|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Profile**  I am an ambitious and hardworking software graduate. I am passionate about my hobbies and enjoy staying busy. I am very punctual and gain satisfaction from completing tasks.  I am open to constructive criticism and strive to complete all tasks I am set to the best of my ability. I have a good sense of initiative and am always on the lookout for ways I can help and contribute. Contact **PHONE:**  +353 89 946 9547  **EMAIL:**  lorcanturner@gmail.com  **PORTFOLIO:**  [**lopyxel.github.io**](https://lopyxel.github.io/LorcansPortfolio/Portfolio/index.html) Main Hobbies Music Production  Game Development  Keeping Active (running, cycling)  Playing and learning new instruments  Learning new languages  Cooking |  | Lorcan Turner StudentEDUCATIONMullingar Community College 2013 - 2018   * 6 subjects including: Higher level Biology (H4) and Construction Studies (H3), along with 3 other higher-level subjects. * 6th year prefect * Attendance award for four years of secondary school * Academic award 2014 and 2015  Technological University of the Shannon 2018 - 2022   * GPA of 77.57 in my third year * GPA of 71.5 in my final year   Result: First class honours degree software and Game development WORK EXPERIENCEThe Learning Lab [Retail Assistant] One week in February of 2017   * Organizing the store * Serving customers * Rearranging shelves * Unpacking deliveries * General cleaning  Mullingar Community College [cleaner/ maintenance]  6th June – 22nd June   * Cleaning classrooms: washing walls, mopping floors, vacuuming, cleaning toilets * Moving furniture * Managing book returns and disposals * Dish washing  Spar (Tills) Summer Fleadh Cheoil 2022   * Lotto tickets and cigarettes, nicotine products and alcohol service * Filling shelves with stock * Cleaning and organising   **Realsim [software engineer and 3d modeler]**  Jan 19th, 2021 – August 26th, 2021 (8 months)   * Planning Applications * Bug fixing, improving, and adding tools to template prefab tools * Documentation, planning, testing * research * Train dispatcher software   PORTFOLIO HERE  References  |  |  | | --- | --- | | **The Learning Lab**  [Magdalen Kelly](https://www.facebook.com/magdalen.kelly.1?__cft__%5b0%5d=AZU6ky145qr8MKuuCNG0AIfDbN-0_qdY4rKTRhxlBCTFMWQWGReHXkQ3zdcios4RP9I8vJBRL2SqOxwEiargB490DHCrMsXv06c0ob38H_X7FvDEweorckC6lJt9FXjg36NLSmgJ3R_aUcRiNGqCXjocIEL1W7s2WkdatjHRpNqZwA&__tn__=-a%5d-R)  (044) 934 2497  **Spar**  Roy  (353) 87 259 3802 | **Mullingar Youth Project**  Catherine McEntee  (353) 4493 49636  **Realsim**  Gavin Duffy  (353) 86 396 2245 | |

**BSC Software Design (Game) Grades**

**Subject Grade**

**1st year :** Mathematics for Software Design 1 49

Communications 50

Digital Media 81

Web Development 1 52

Software Development 1 (java) 49

Agile Methodologies 50

Game Development 42

Computer Applications 56

**2nd year:**

Databases 2.1 40

Game Development 2.1 (C#) 60

Software Development 2 (java) 61

Agile Methodologies 2 51

Networks 75

Group Project 64

Software Development for game (C++) 72

Mathematics for Software Design 2 61

Game Development 2.2 (C#) 77

**3nd year:**

Pending

**Projects:**

**(link to current game project)**

[**https://www.youtube.com/watch?v=2zRB3kyvAmA&feature=youtu.be**](https://www.youtube.com/watch?v=2zRB3kyvAmA&feature=youtu.be)

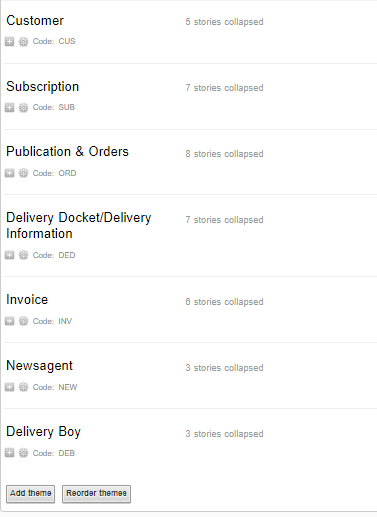
**Publication management project in an agile SCRUM team (Group Project)**

The product we intended to build was a system designed for a newsagent allowing them to keep track of customers, publications, deliveries, and orders within the business. This system would not only be easy to manage and control, but ideally easy to learn how to use. There is always a learning curve when learning to use a new software.

This system would store data into a database attaining all aspects of a newsagent’s business. I.e. customers information, what subscription type they require, how often they receive deliveries, whether the customer has payed or not etc.

We also wanted to allow the ability to remove customers, orders, publications and add new ones. Even managing the delivery boy log in and check in times. Having documents printed automatically we really wanted a system that could manage everything. Therefore, the project was named Publication Management.

As a team we had to decide the Entities of Publication Management. We had roughly done already by splitting up the user stories into which Entity the fit into.

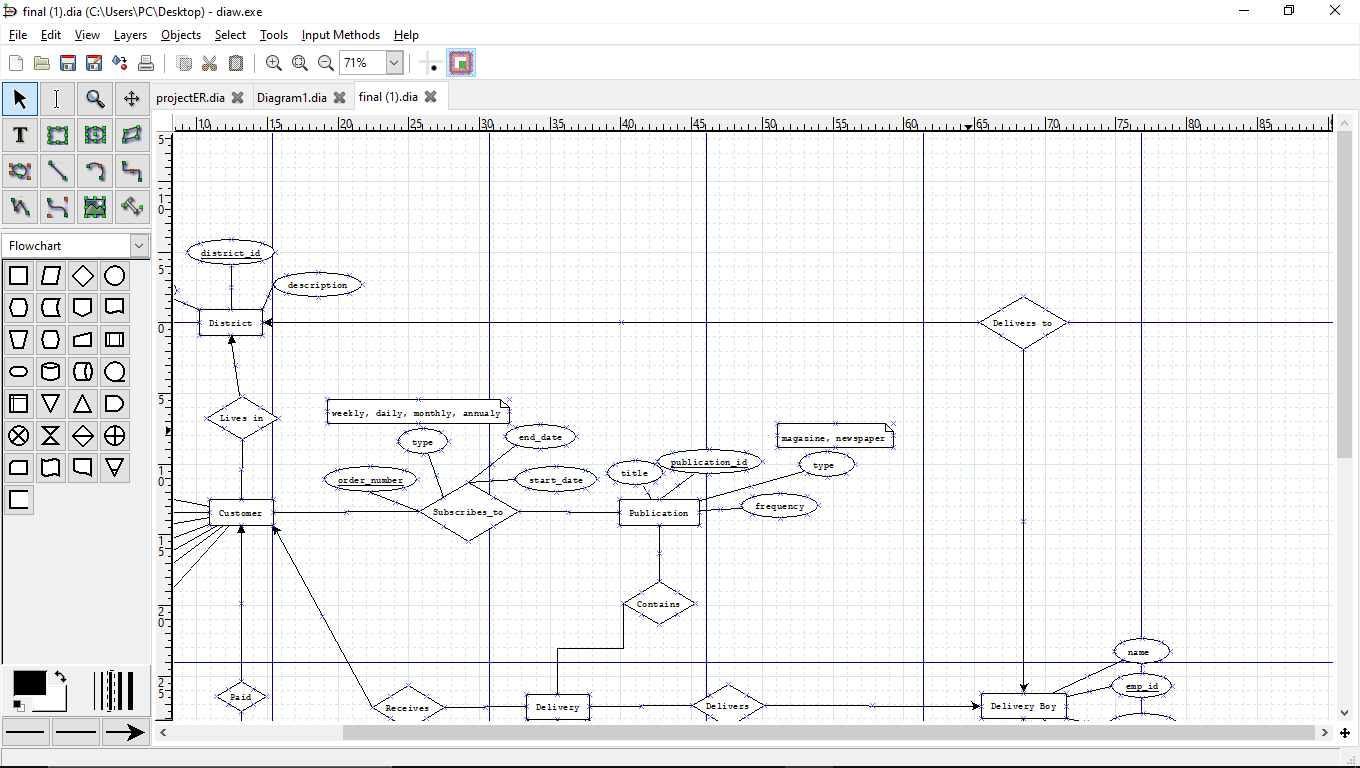


When designing the ER diagram, we connected Entities by using relationships. We also realised the subscription section was a relationship and not an Entity because a customer subscribes to publication.

We had later added our own new relationships and made new Entities based on individual user stories. For example:

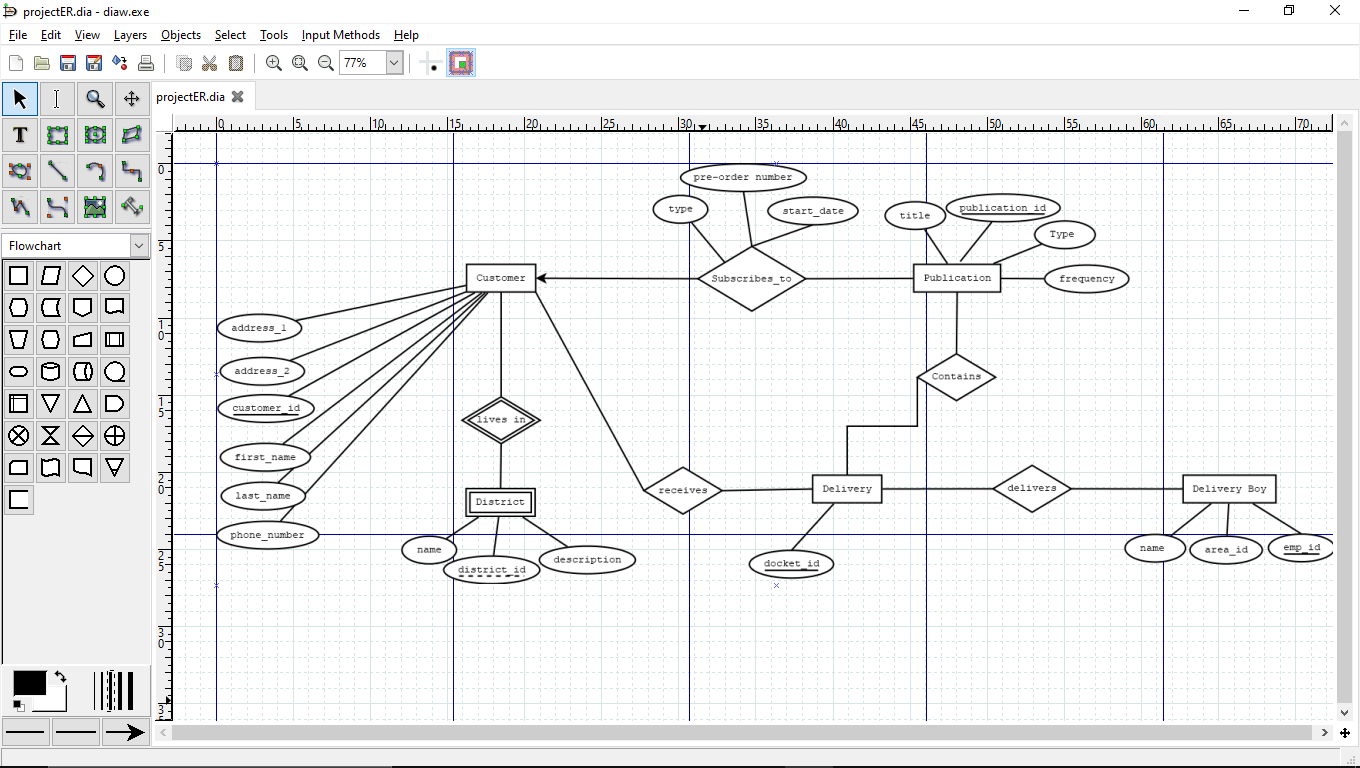
A customer lives in a district. Therefore lives\_in is the relationship.

A delivery boy delivers to a district. Therefore, Delivers\_to is the relationship.

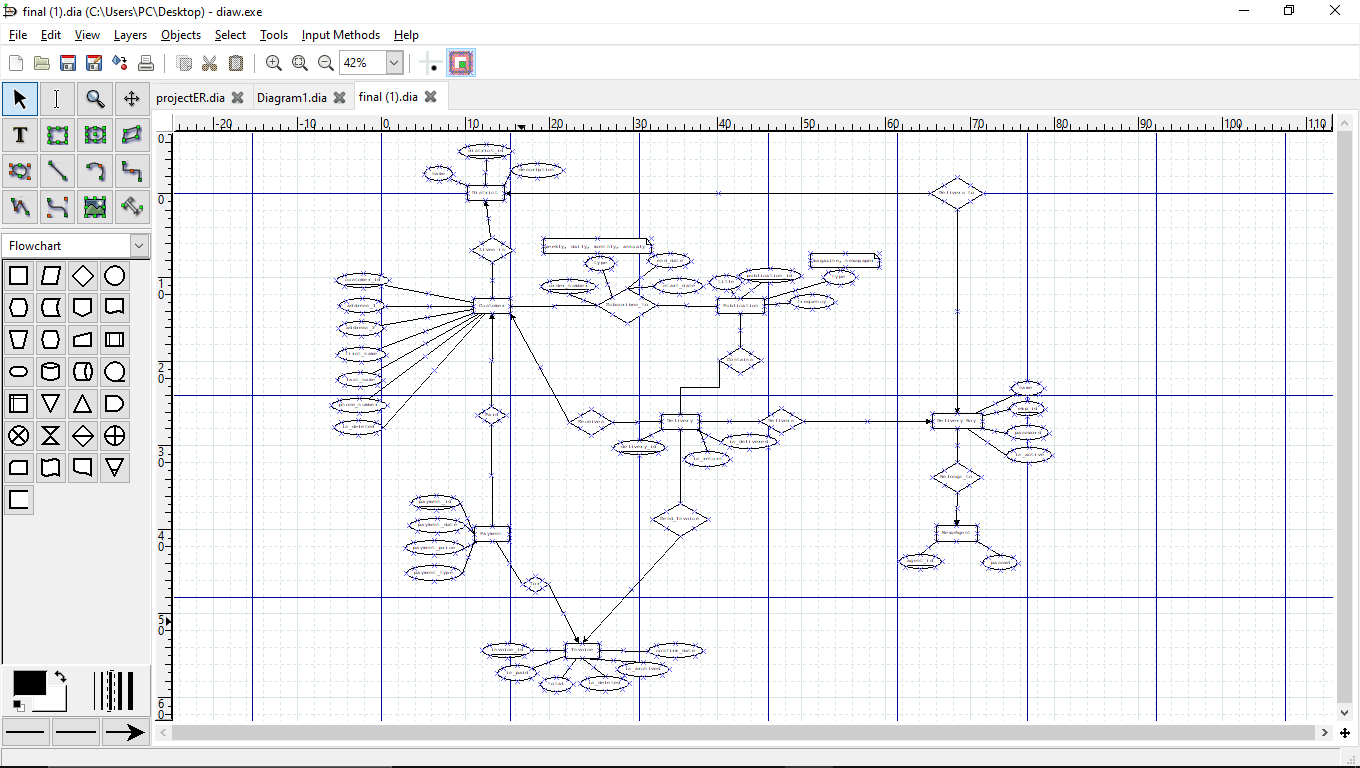


My initial attempt at an ER diagram turned out good but was still missing a lot of information.

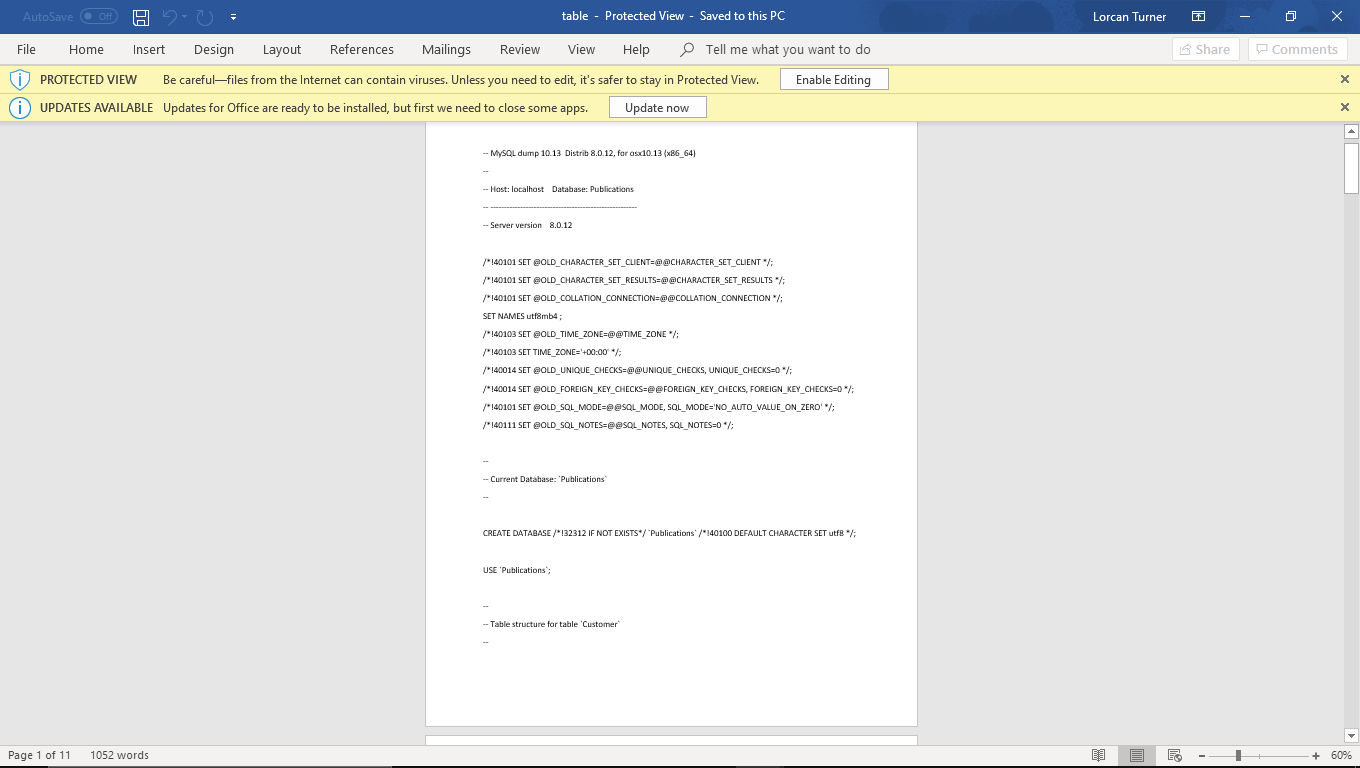
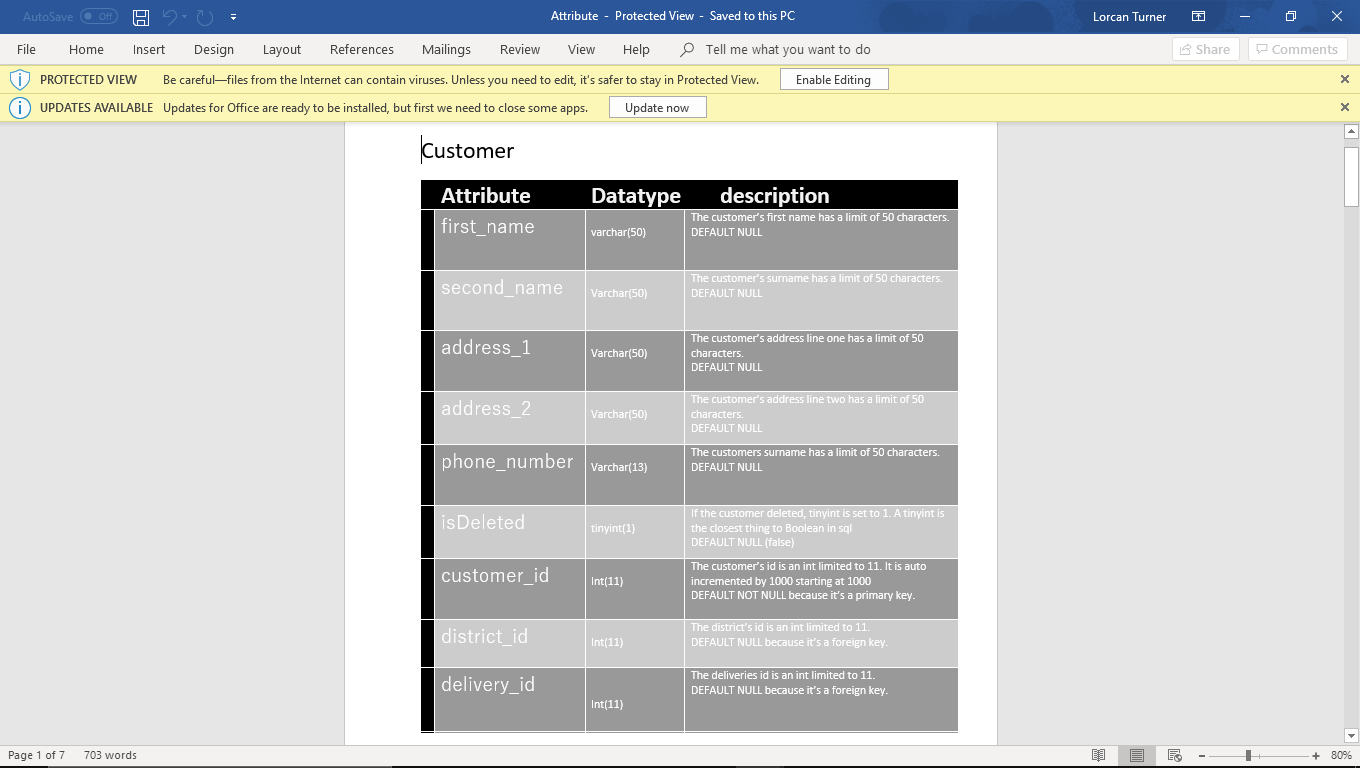
In the later weeks when our team was put together, we looked at every member’s diagram attempts and put best elements together.

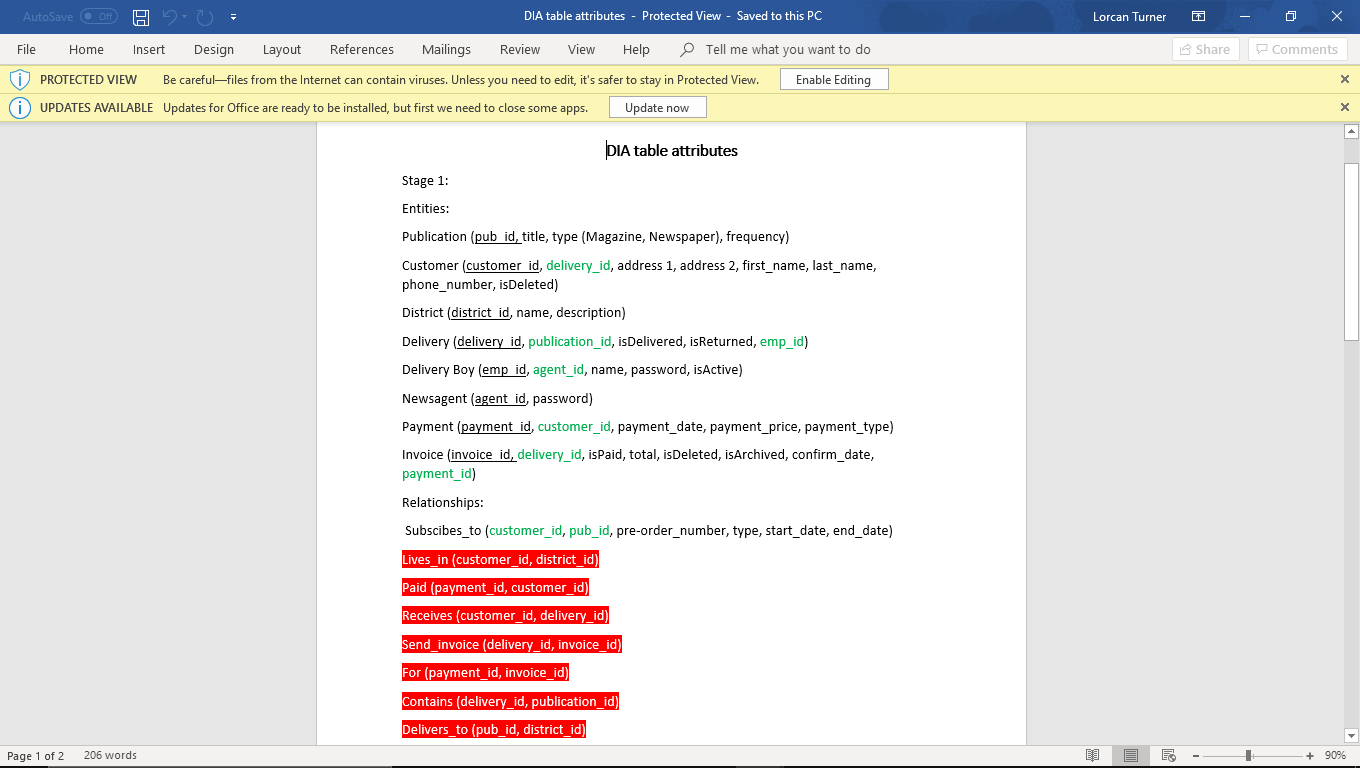


After reducing, tweaking and discussion our diagram still ended up being quite large. It was neat, easy to understand, left with little descriptions over the attributes that needed clarification. The type of relationship the Entity had with other entities were indicated with arrows.

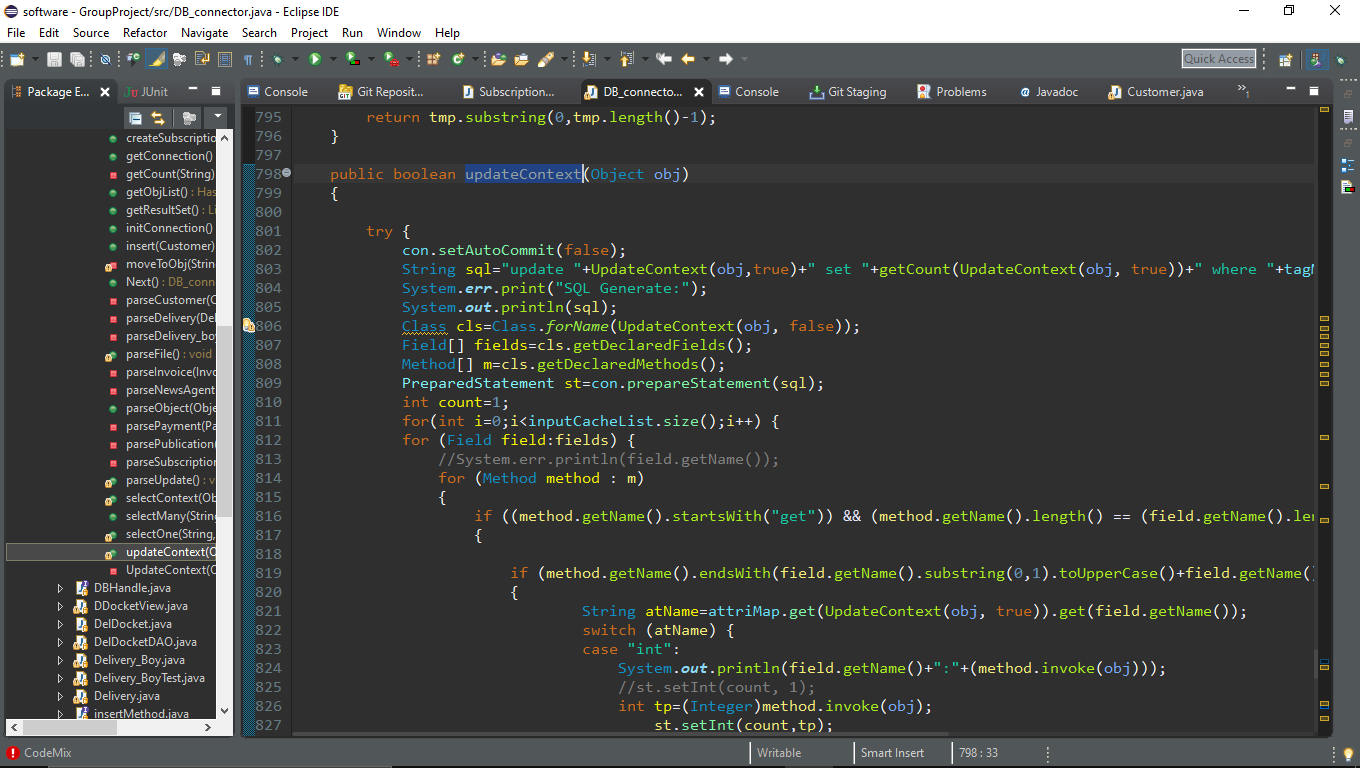


When converting this ER most if not all the database code within the project itself was written by one member. During Group project classes we did much verifying. We had copied the code onto the Microsoft “work together” app on the browser allowing changes to be made by everyone live while I called out each entity and attribute as well as the datatypes required, default values and limits.



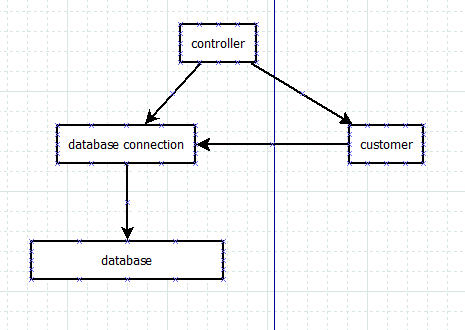


The way in which we had access to the code in our classes was by calling specific methods they created. For example db.UpdateContext(customer, cus\_id) would update a row in the table containing the id value left in the customer class.



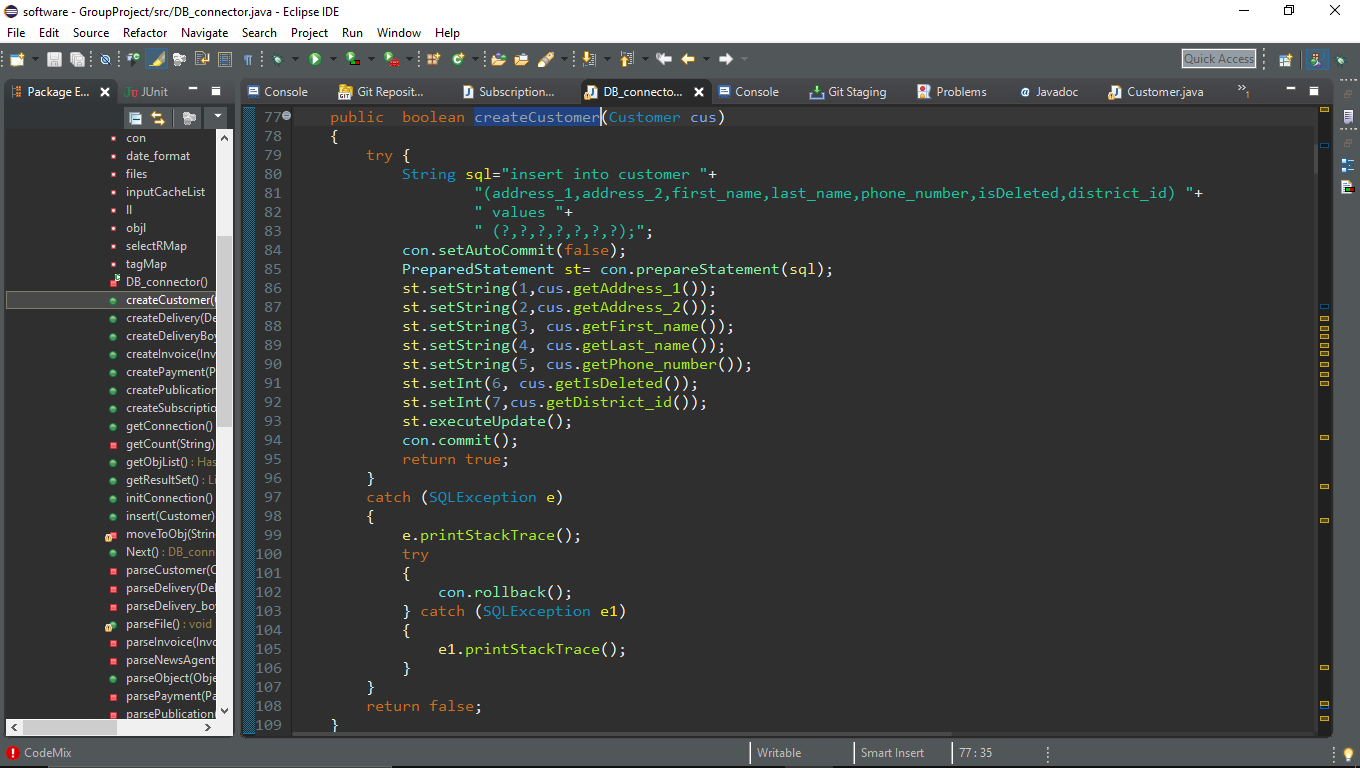
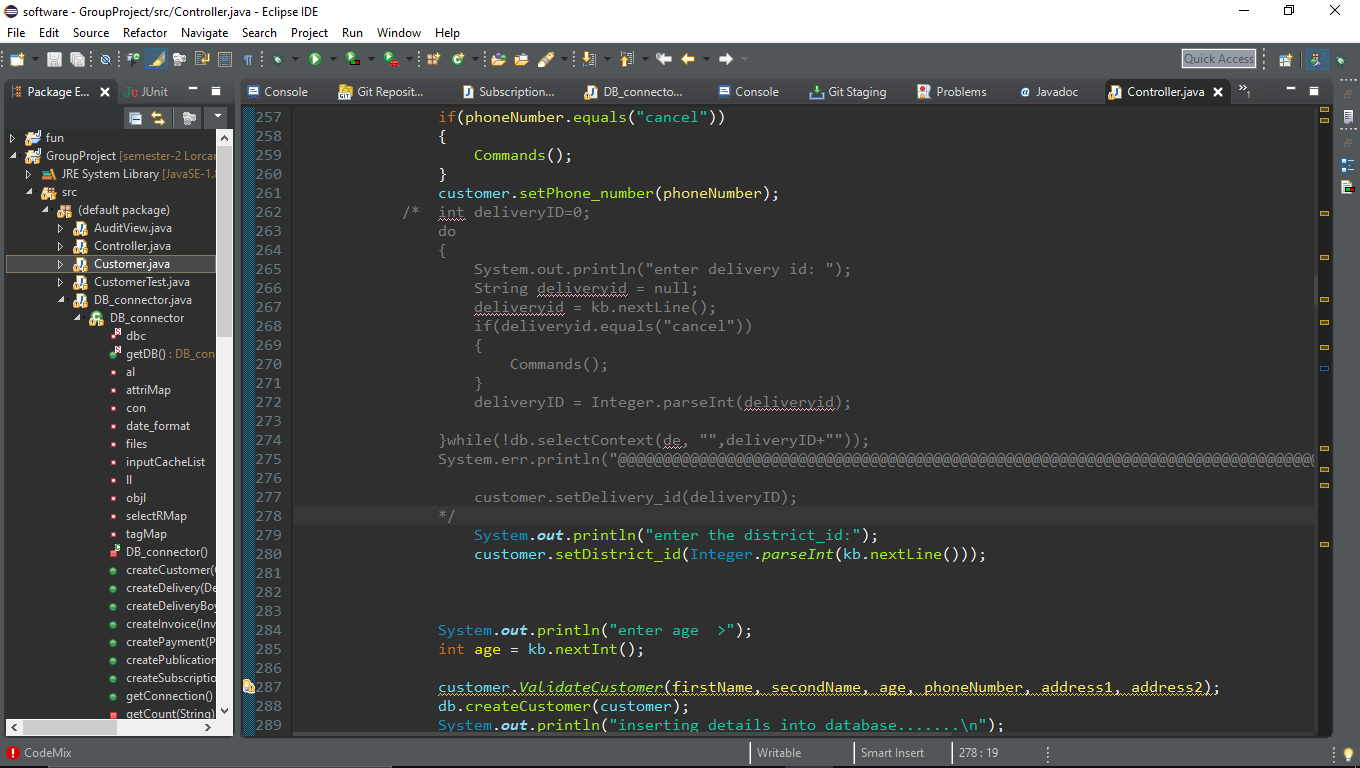
Architecture:

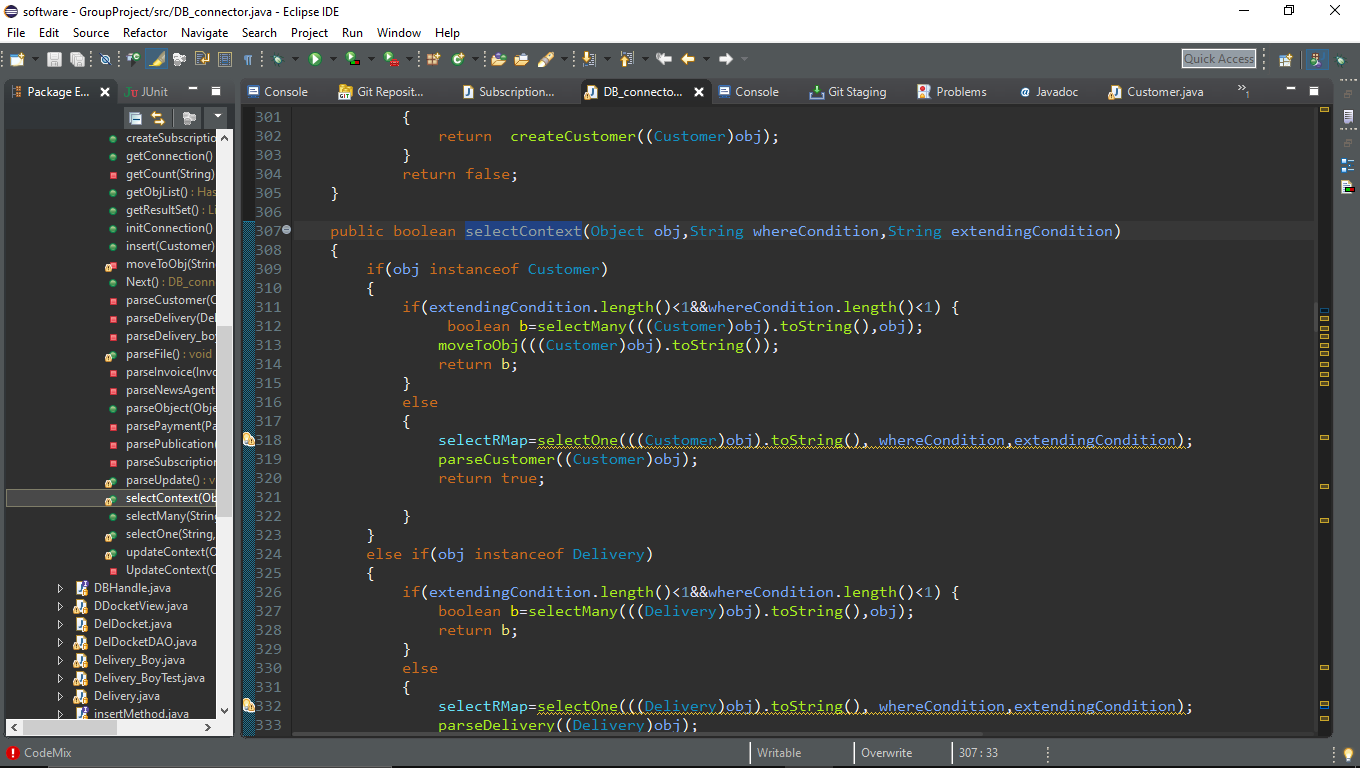
Having already started code, our plan went from a load of classes filling up and not communicating, to an MVC like architecture. We had decided on this as it would have allowed us to create and implement a GUI into the project much easier. Instead, the way we ended up carrying out this architecture caused one massive controller class which communicated with most methods via the command line. In hindsight I would try implement MVC better so less methods would be lost in command line code. The command line code being put into the controller was a very messy way of doing it as it had nearly 1000 lines of code which made looking for methods more confusing. The Small bit of GUI our project did have though, did have a view class that was much better kept.

Customer Example:

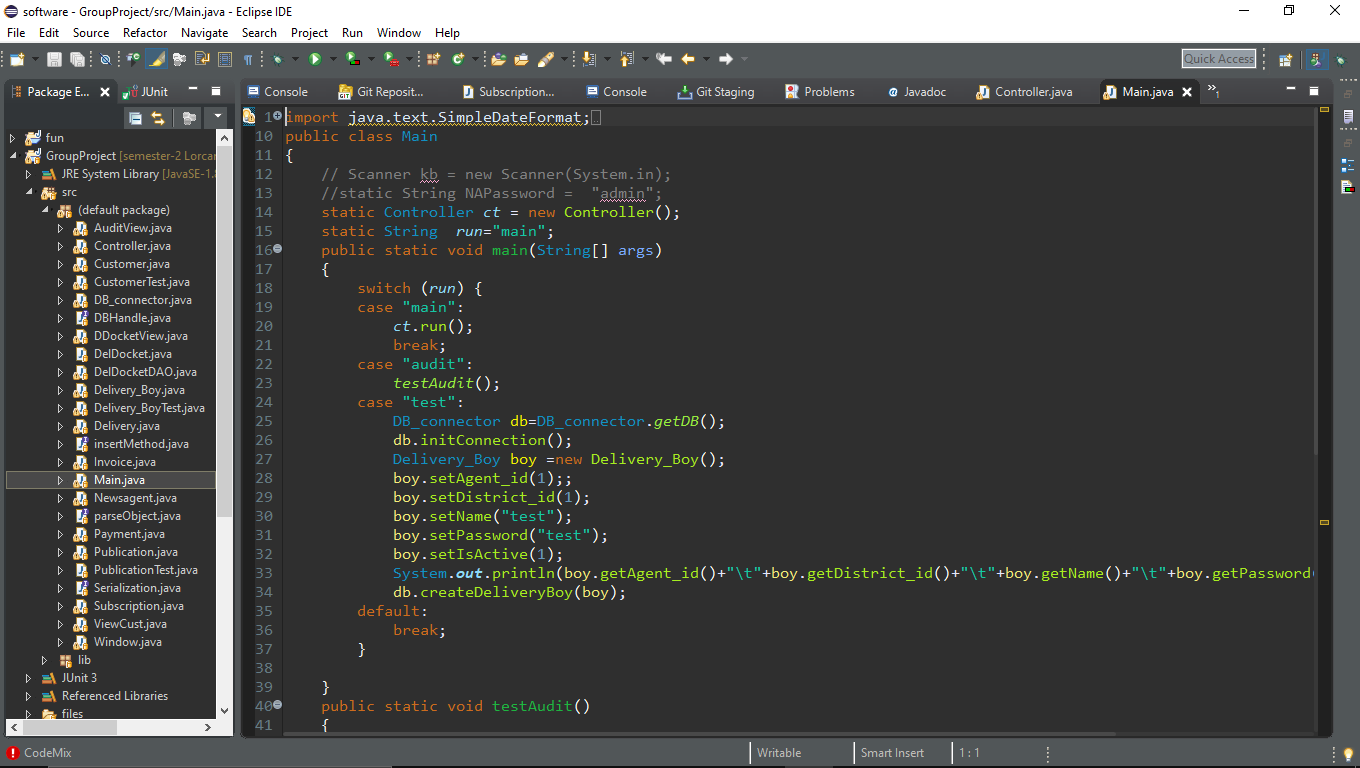
User details are inserted into the controller. These are passed into the customerCreate method which sends details to the customer class.

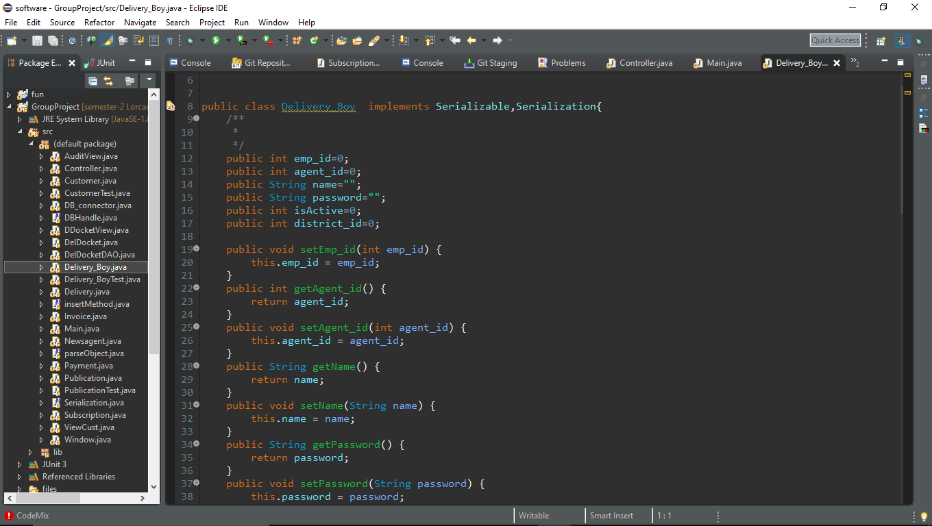
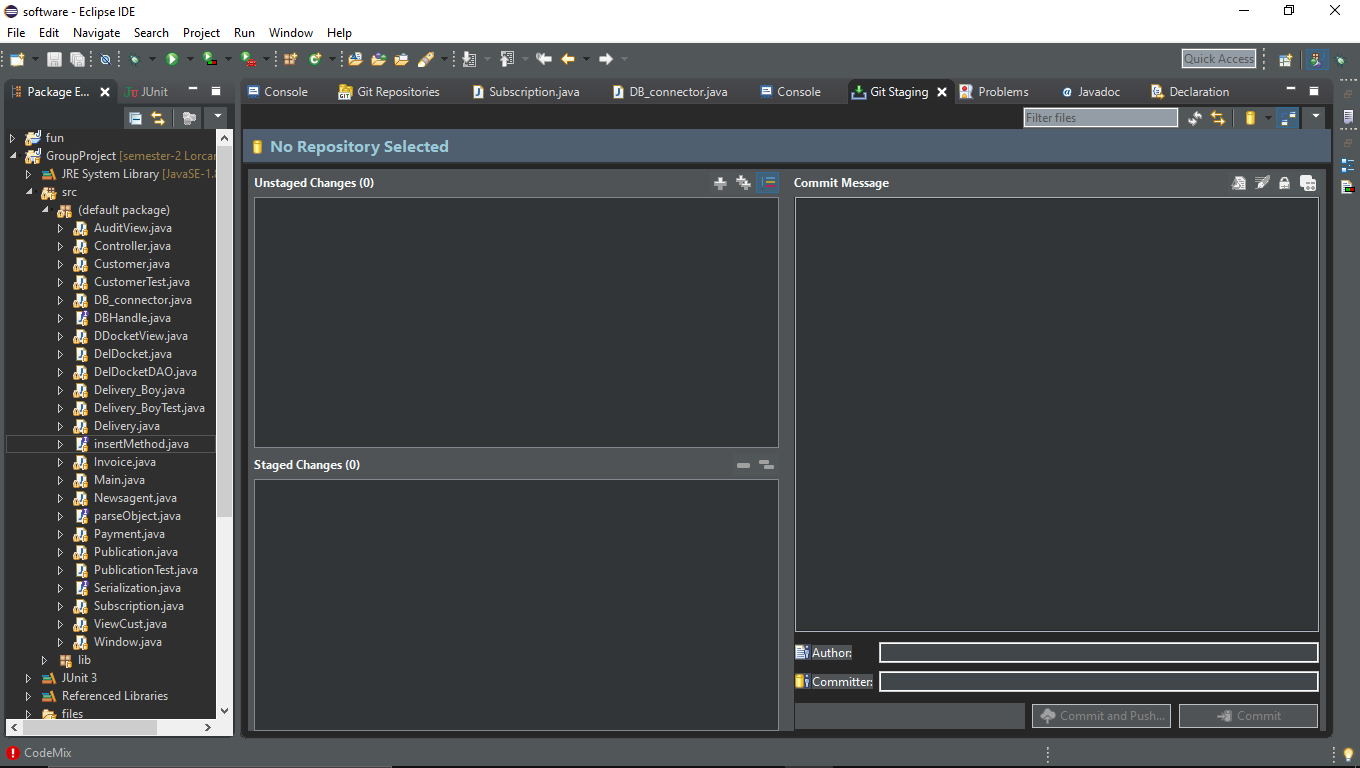
In the same method the database connection method called customerCreate is called looking for the parameter of an object. The customer object with all the inserted details is put into the connection method which contains database insertion code. The customer is then created.

As shown here

other database classes were used like this to allow us to view an entity when passing in an object as well as an ID.

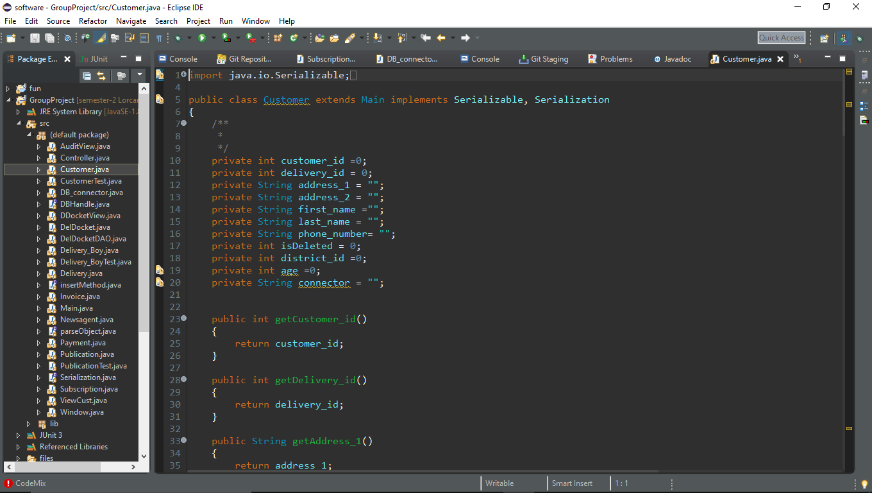
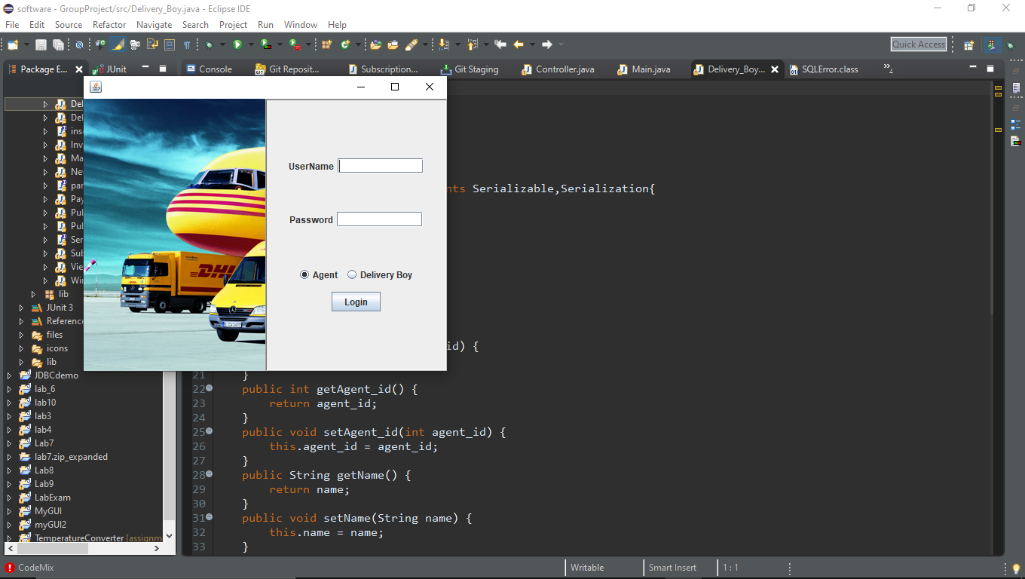
Another architecture decision we made towards the end of the project was to use a switch statement to allow us as developers to work form a starting point rather than retyping in the same details whenever we wanted to test our program.





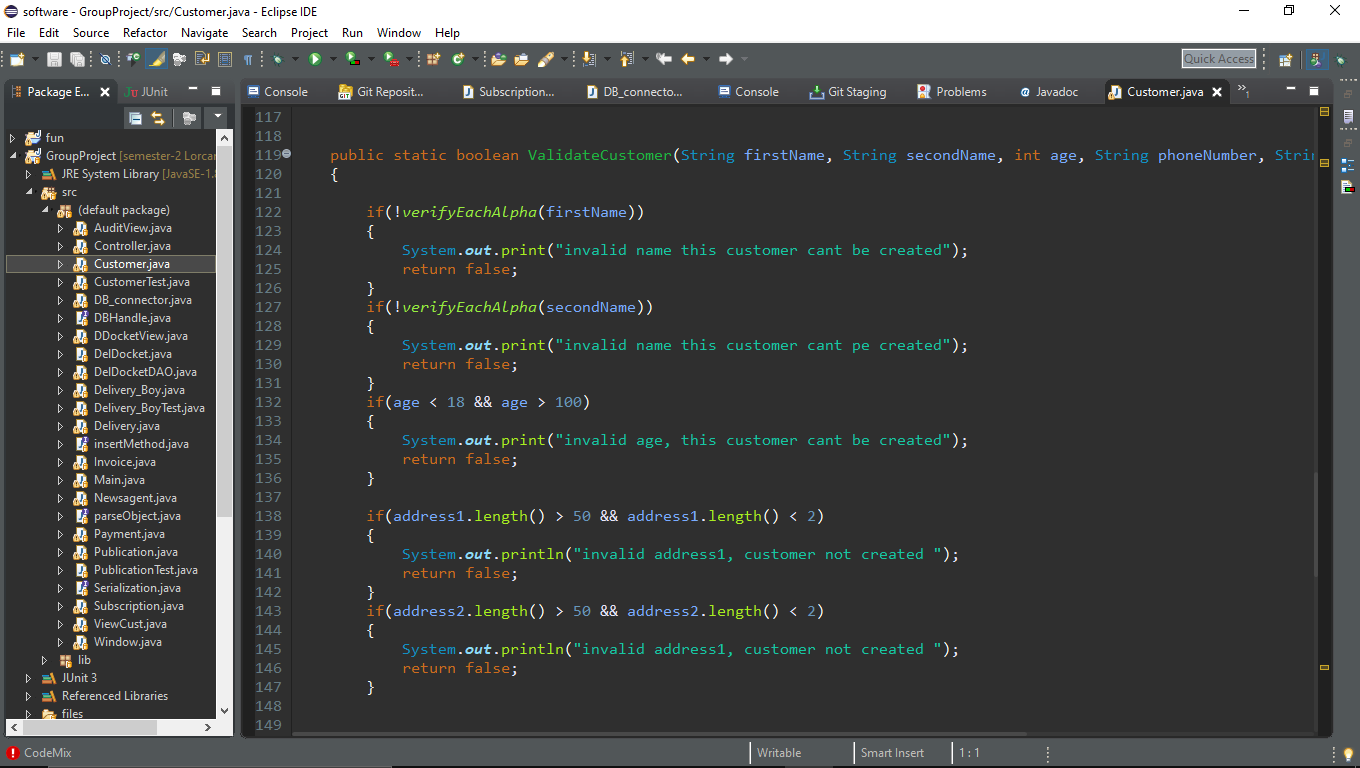
The classes I created were the customer class, Delivery boy class and the controller class. It was my job to maintain the controller classes allowing classes to communicate with the command line.

I had to communicate the buttons in the GUI to send the delivery boy to the delivery boy menu and the new agent to the newsagent menu depending on who logs into the system.

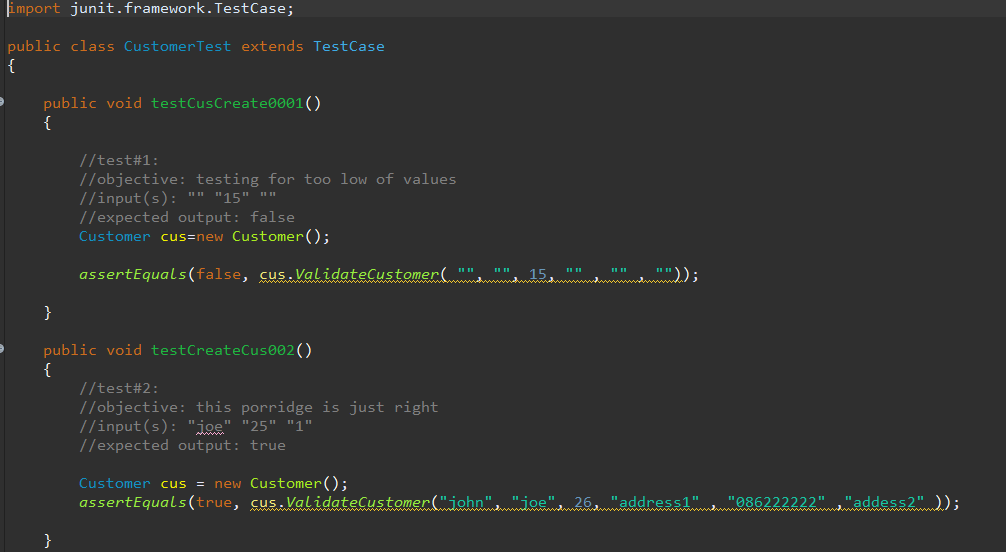


Within the controller class I created the first create view and modify methods. I asked the members of the team to copy this code and to use the same format in my methods as well as their own values. Ie. createCustomer() createPublication(), createSubscription()

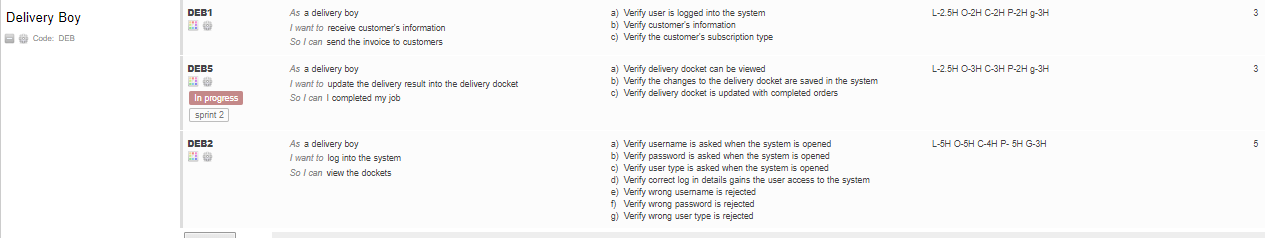
This made many methods which made the controller class too long.



For testing the code, I created a method within the class. For the customer class for example ValidateCustomer was created. This would be called in the assertEquals that takes a Boolean as the output/return value and the entered details as the input. In the product code if this class is false, you will be asked to re-enter details.



Here I am testing for values in and out of the boundary of validity.

When the user stories were first written we had designated a colour per person, the stories since then have been reworked so many times that who did what is a blur. One of the last edited was for the log in. 

On the very right are time estimations. As a group we all individual went through each user story verifying them and estimating how long we thought that user story would take to do.

If I was to do a project like this again, I would work strait form GUI and ditch command line. I think this would have put our software studies into practice as well as compressed the controller class.

More team communication would have created a better understanding for everyone of what was going on and what need to be done.

All in all, this project thought me a lot about working in a team and the many challenges a team may face. I was very lucky to have members on my team who would not only help me but listen when I had bits of help that I thought I could offer. I was more limited in my ability to code before the project than I was after the project. Even though by the end, it wasn’t the initial vision, it was a good learning experience.

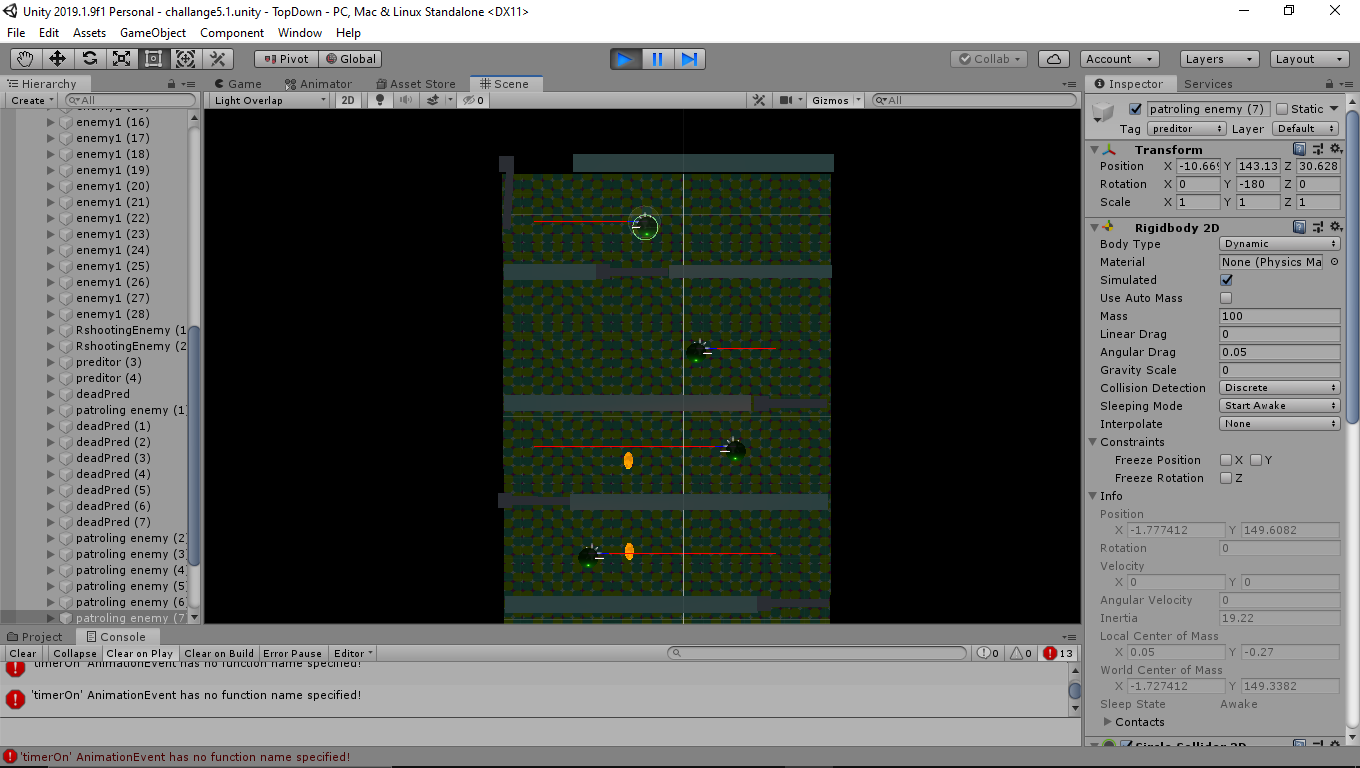
**Game Artificial intelligence AI(individual project, GameDev 2.2)**

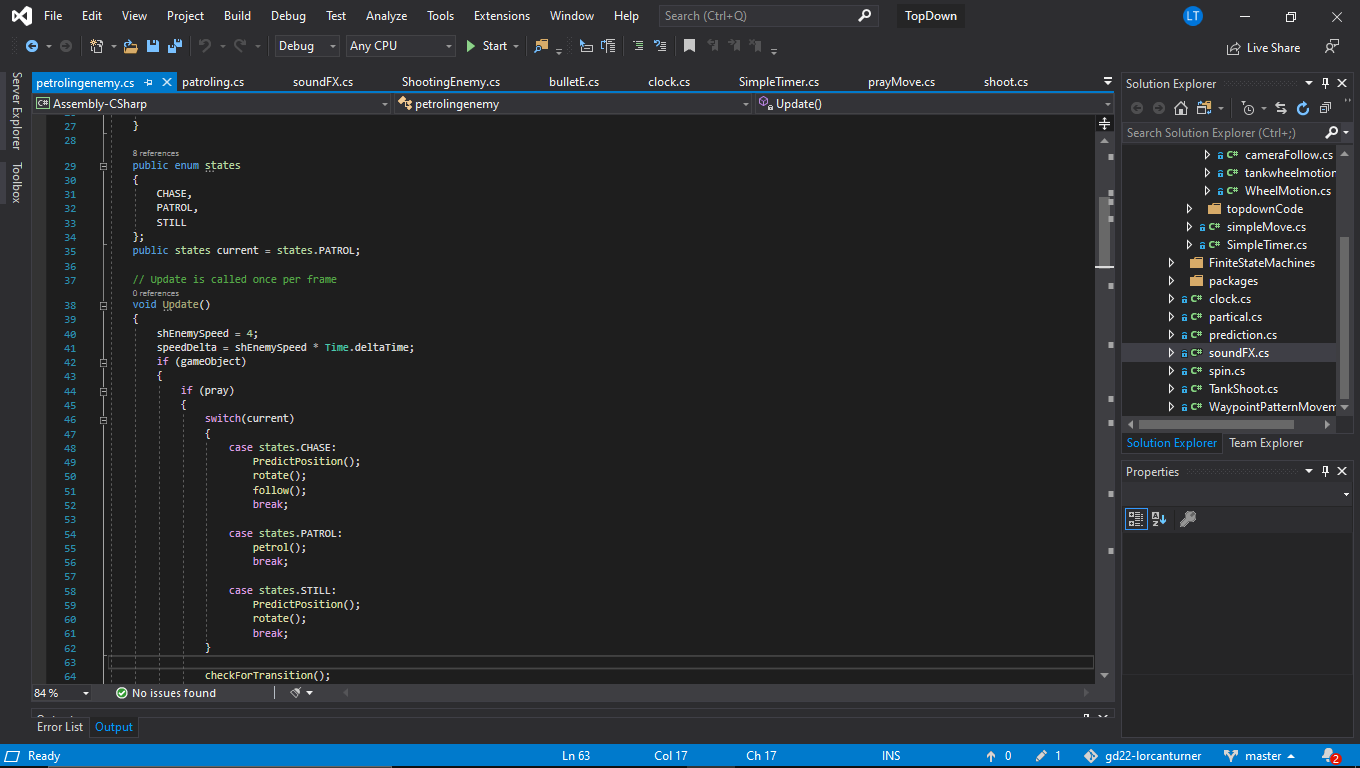
Game Development 2.2 TopDown

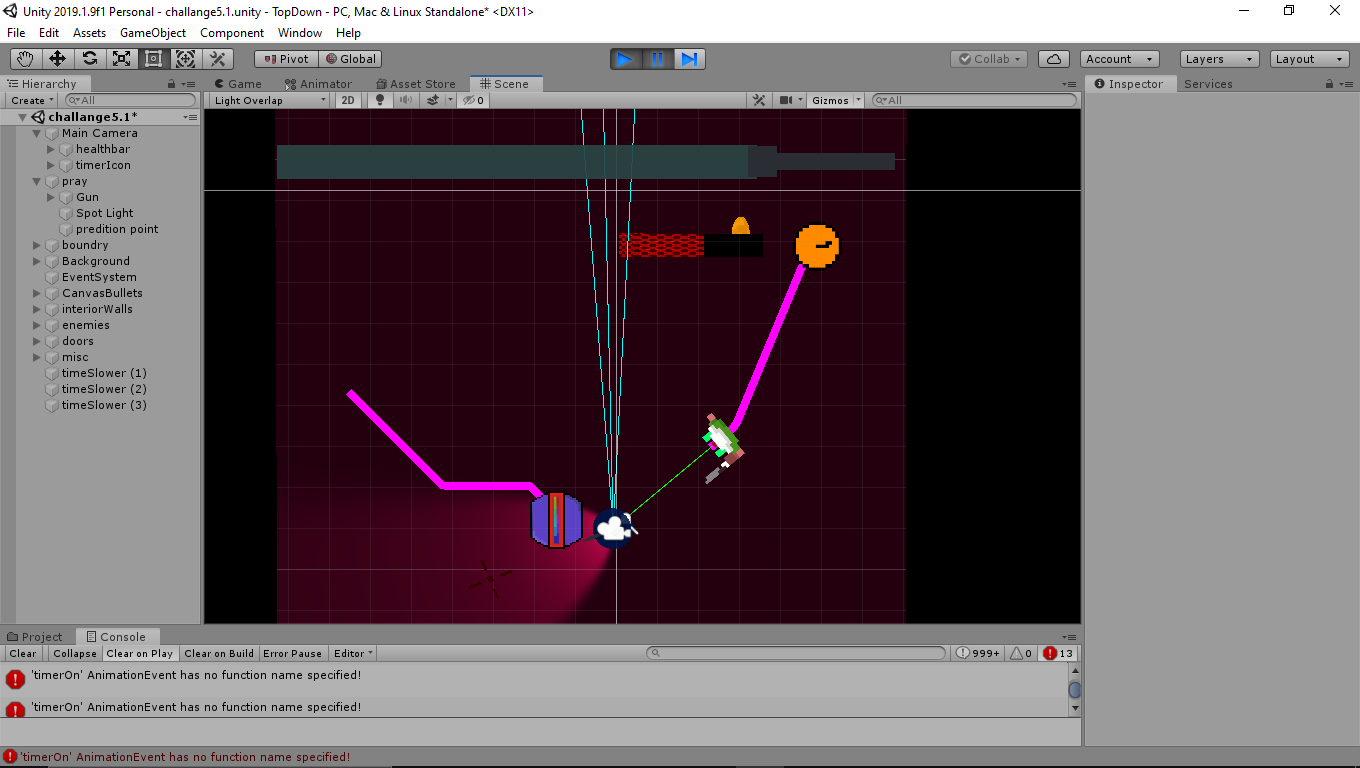
From challenge one to five I have been working on a top down shooter where every kill counts. Dodging enemy bullets, running away from chasing, patrolling and predicting enemies as well as picking up health, keys and slowing down time, lot of elements learned throughout the year have been utilised well to make a game AI demo.

**objective:** kill all the npc's in the level to move on to the next one. 5 bullets in a mag reload quick or die slow down time refill health the keep the clowns away

**Gameplay:** within the game there are 4 types of enemies. the clowns only follow when you are at half health, the purple balls spin with blades when damaged, the red shooter follows at a certain distance and will stop a certain distance away to allow it to shoot and the patrolling enemy walks a path until the player is within a range before it chases and shoots. keys allow you to open locked doors health increases your health by 1 the clocks slow down time for a limited time allowing you to dodge enemy bullets or increase accuracy

**Pattern Movement:** Within the game there is a patrolling enemy. This enemy has the same abilities as the shooting enemy except this enemy when not within the range of the enemy, will walk a designated path. This path can be edited whatever way is best for the level design. [](https://github.com/aitsoftwaredesign/gd22-lorcanturner/blob/master/Images/Screenshot%20(12).png)

**Finite State Machines** : finite state machines are a great way of giving you character multiple states of being in a way that’s easy to read, code and manipulate. fsm is a code configuration using a switch statement. states can change under specific circumstances. ie. if distance is < 4 change the state to chase. [](https://github.com/aitsoftwaredesign/gd22-lorcanturner/blob/master/Images/Screenshot%20(8).png)

**[](https://github.com/aitsoftwaredesign/gd22-lorcanturner/blob/master/Images/Screenshot%20(9).png)line of sight chasing:** we did this in multiple ways. in the basic chase we simply said if your less than the x position increase, if your greater than x position, decrease. The same was done for the y axis. this works but has a few flaws such as jittering and moving indirectly. I kept this chase for the basic enemy to allow for movement variation in my game between enemies. For the rest of the enemies the "moveTowards" function was used. This is an inbuilt function that allows for a smooth movement from one position to another.

**Vector Math :** Vector math was used in this project for calculating and transforming positions and rotations of npc's in relation to themselves and the playercharacter. For instance the shooting enemy when he is out of range is stationary, but when the player is within range the shooting enemy rotates towards the enemy to shoot towards and transforms its position to a place closer the the player.

**Interception:** Within our project we were set the challenge to use vectors the predict where the player character is going so the enemy can intersect the players path. this was done by using the velocity of the player to determine the normalized vector for the enemy to follow.