Team 35 Report

Introduction

The purpose of this report is to present our planning, designs and our query processing. The report will also include an evaluation, where we assess how well we have worked overall, and what changes could have been made.

We started by meeting up to review the scenario and evaluating the requirements. From there, we set ourselves the deadline to finish the diagrams within a week to leave as much time as possible for creating the software. In addition, we decided that after the diagrams were completed we would separate the work (implementation) between us and meet at least once a week. At each meeting, we discussed what we achieved individually so far, and what we planned on doing in the following week.

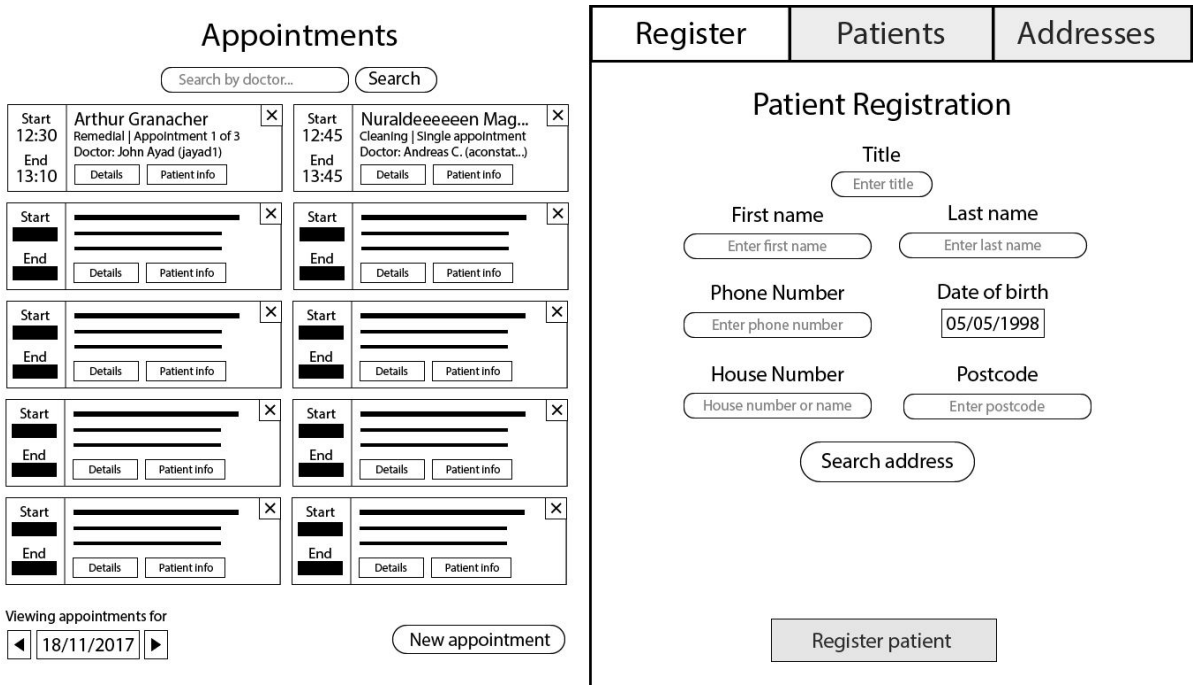
Before creating the software, we decided to separate the work using the UML information model, so that three of us would distribute the classes between us and their corresponding database tables. The last team member worked solely on the interface of the software. In the case of new work arising, we would assign the work to whose classes relate to it the most or whoever could handle the extra work. Every member was responsible for the testing of their own classes.

Design

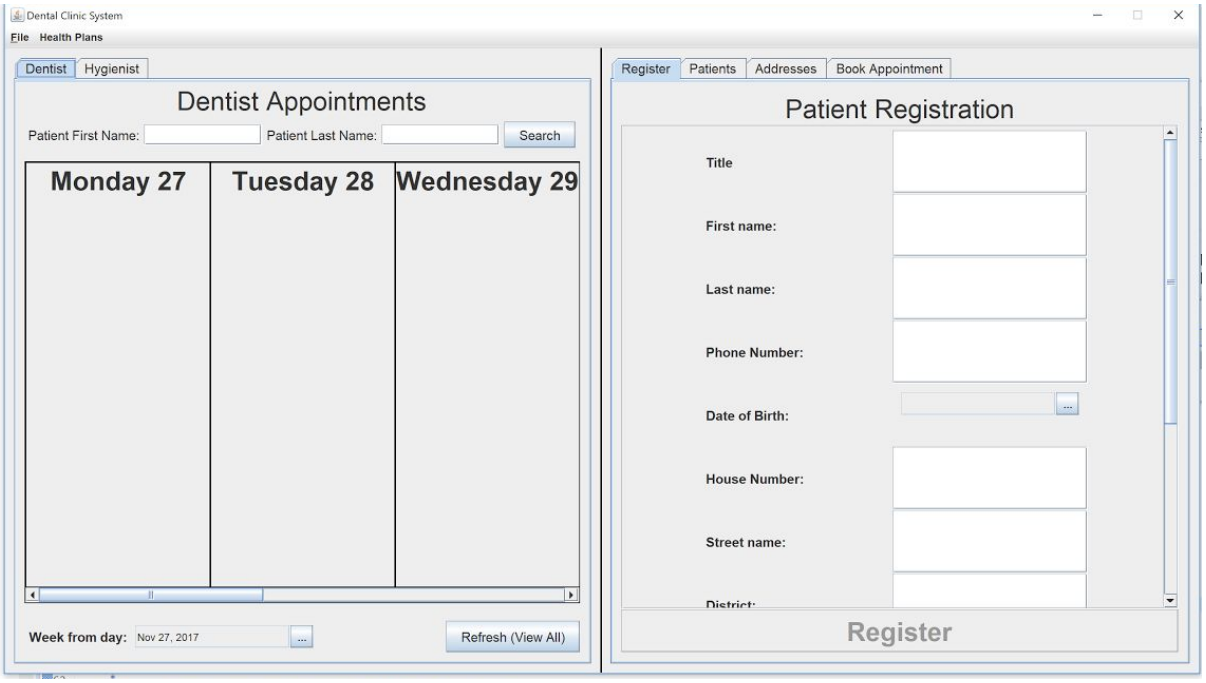
Before committing to writing any code for the GUI, we decided it would be best to draw mockups of our design to solidify what we needed to include, and most importantly, how we were going to present the functionality most effectively.

Secretary View

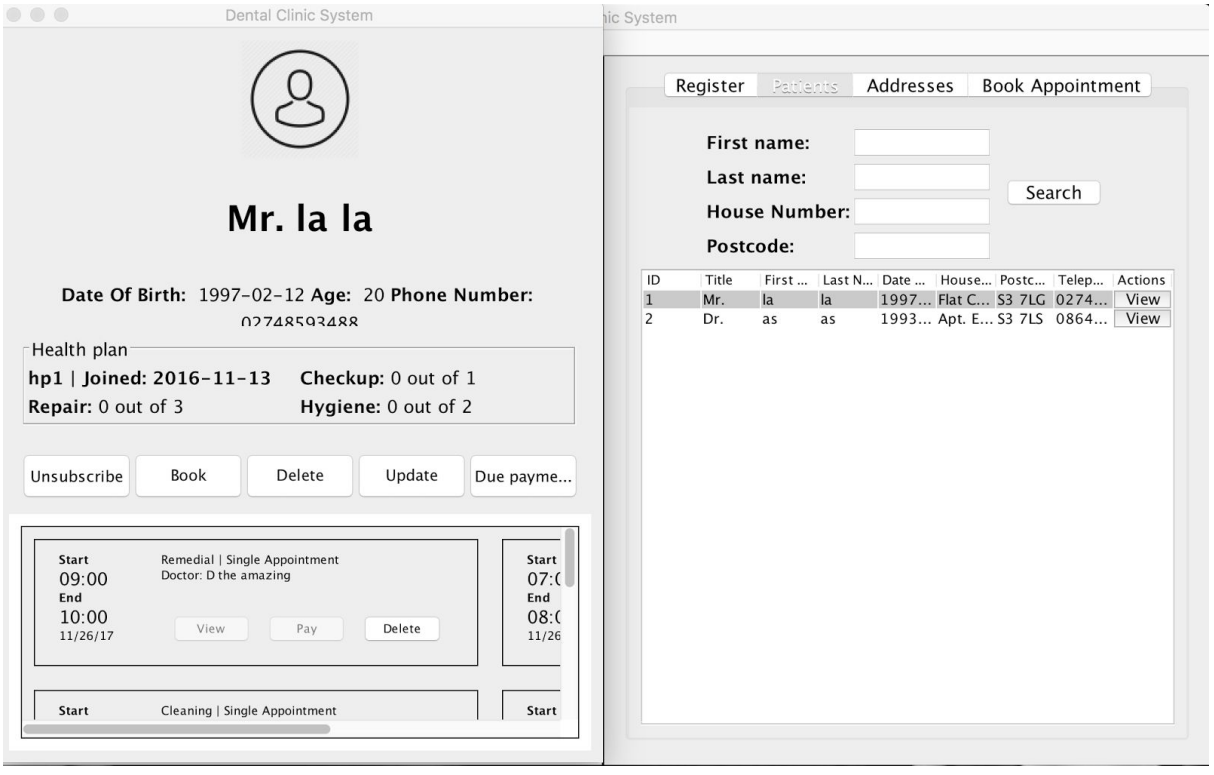
The secretary view required a lot of thought to design due to the number of different tasks that the secretary needed to be able to do. It was important for them to be able to view varying information simultaneously. For this reason, we decided to design it as a split-view window. On the left-hand side, you can view all of the appointments by day and narrow them down by doctor. Since this is something that the secretary will be referring to very frequently, we thought it best to have that there permanently. On the right-hand side of the window however, we split it into 3 different sections: Register, Patients and Addresses. We could've had a 'Register' button inside the Patients tab, but since registering will be done frequently, we decided to have it accessible in one click instead. The Patients and Addresses tabs are used for searching for patients and addresses respectively.



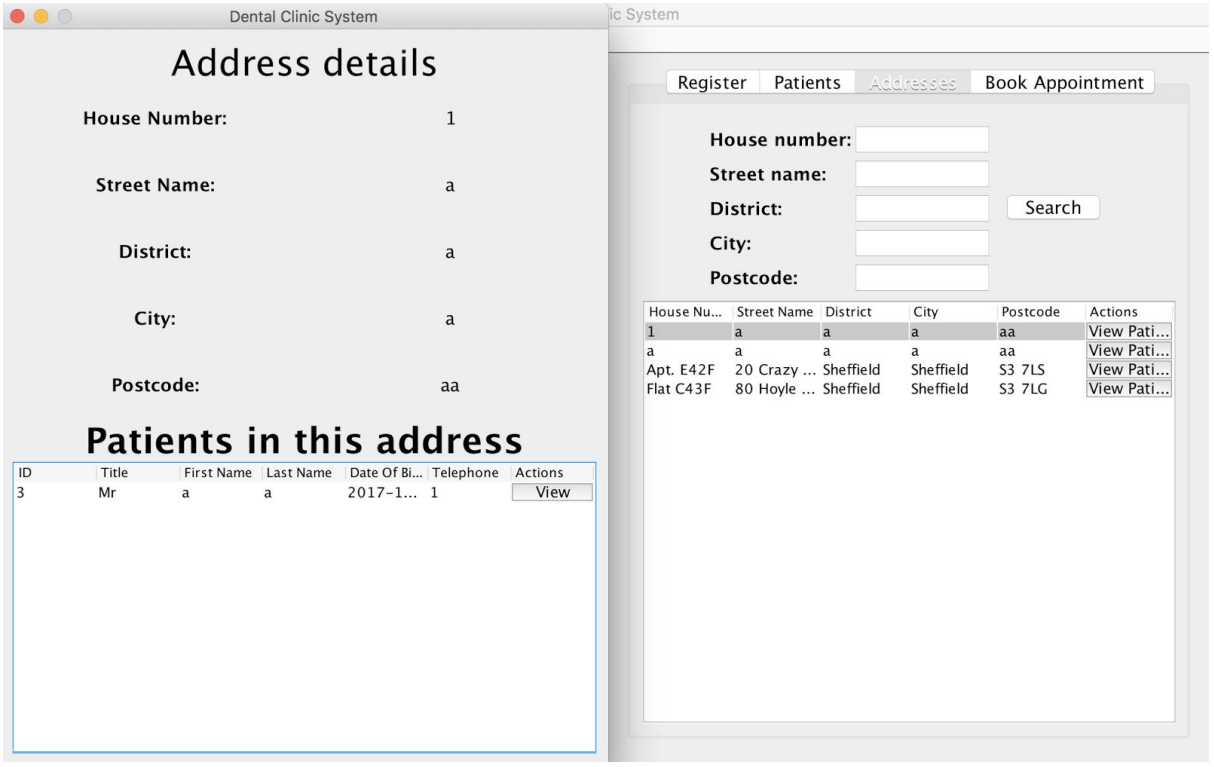
The final design for the secretary view differed slightly from the planned design. The major design change was having two tabs for each doctor for the secretary to be able to see their appointments separately and in a week-view (see below).



Below is a screenshot of the Patients tab and a patient's details, which can be seen when View is clicked on a specific row.



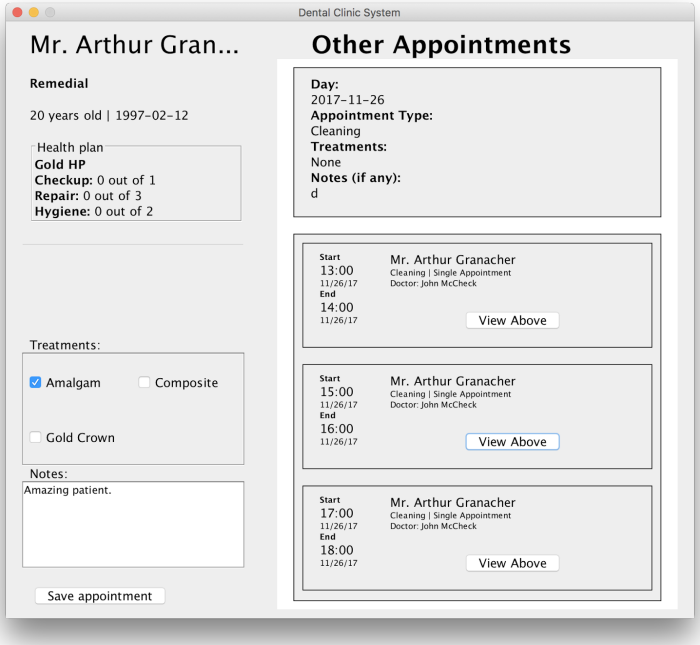
Here you can see the Addresses tab along with the details of a specific address, which, again, can be seen when View is clicked. The address details view allows the secretary to see a list of all patients at that address.



Doctor's view

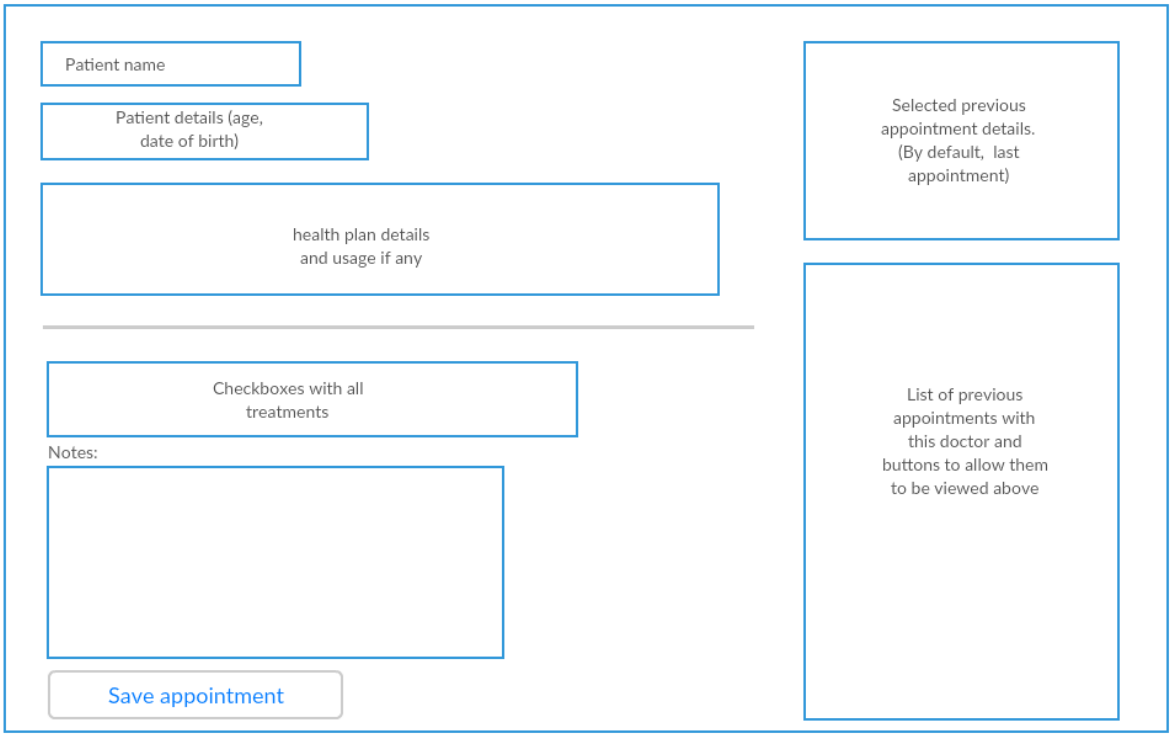
We focused on simplicity when designing the doctor's view since this works best when using touchscreen devices. To achieve this, our system shows only the necessary details, with further information and actions only a tap away. On logging in, the doctor sees the current day's appointments with an option to change the day if needed.

The doctor can choose to view an appointment's details, patient's details and can cancel too.



On tapping View on an appointment, they are displayed with all the relevant information they might need in an appointment. The left hand side displays information for the current appointment along with a list of treatments that can be added to the appointment using the checkboxes. The option to add the treatments which have been done in the appointment is only available if the appointment is of remedial type. On the other side, a list of the patient's previous appointments with this doctor is shown and on top is a small box that shows details for the selected appointment from the list below. This is to help the doctor review any previous notes that might have been left in the any of the patient's previous appointments.

As you can see from the mockup below, the layout of the final design is very similar.



UML Diagrams

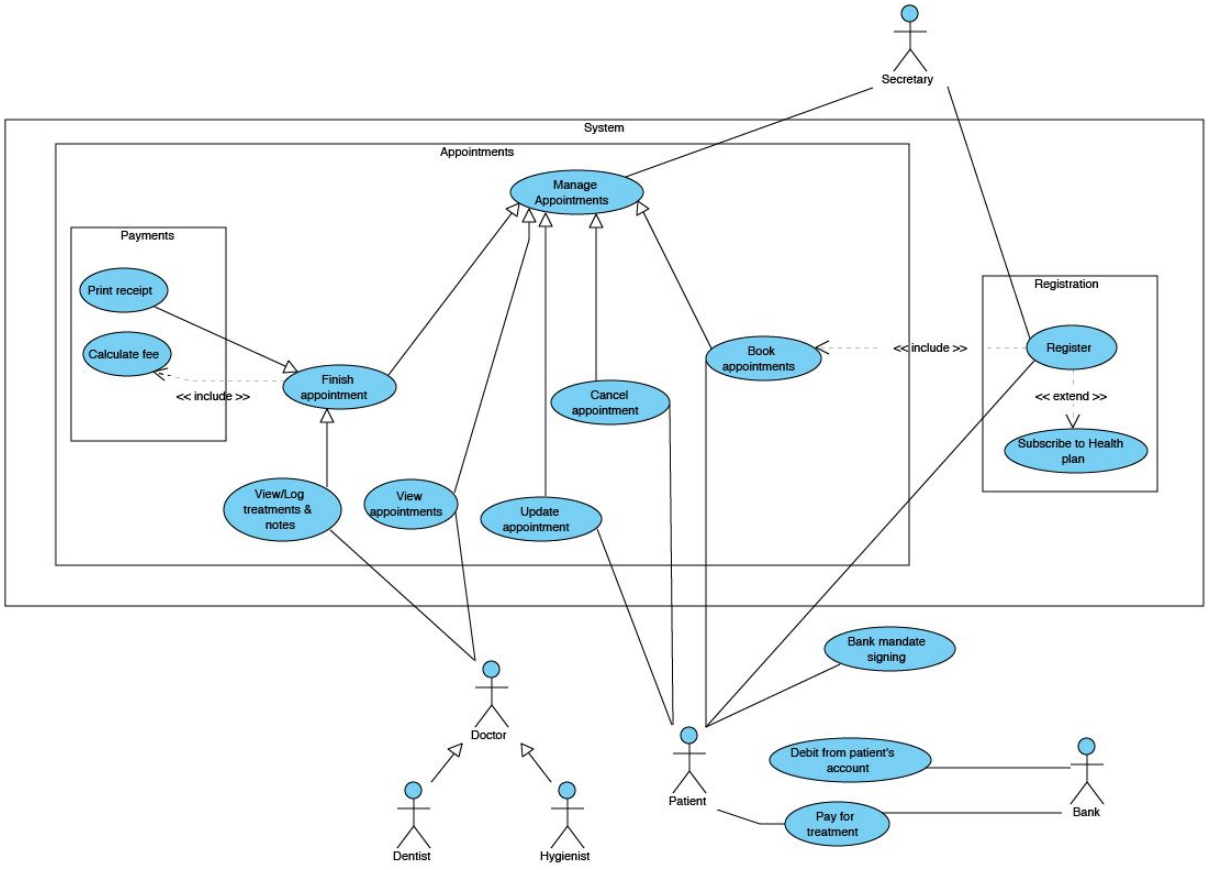
Use Case Diagram

The figure below is our representation of a use case diagram, in which we tried to capture the relevant information from the scenario.
As you can see, we chose to generalize the dentist and hygienist as a doctor (actor hierarchy) because they carry out the same tasks. However the secretary and patients' tasks are separate to the doctors, so they are their own actors.

In addition, we decided to include finish appointment when dealing with calculate fee, as the appointment should be finished before it is to be paid. We also decided to include booking appointments when registering a new patient, as whenever a new patient is registered an appointment with both doctors should be booked.

Furthermore, we used a use case hierarchy for managing appointments where book, cancel, update, view and finish appointment are all similar tasks that fall under managing appointments.

Lastly we have chosen that subscription should extend registration as the patient may or may not choose to subscribe to a health plan.



Information Model Diagram

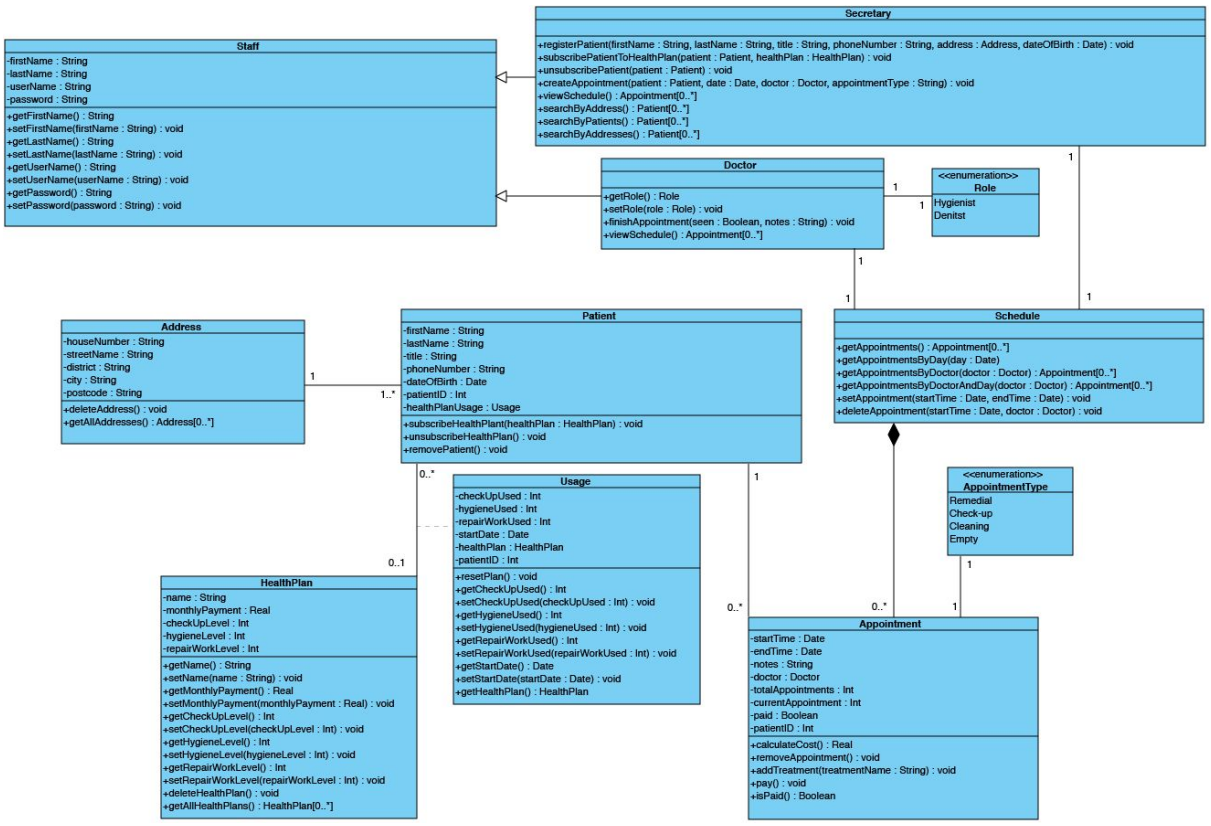
The figure below is our representation of an information model, in which we tried to capture the relevant information from the scenario.

Patients have their own class, with attributes of their details. It's methods allow a patient to subscribe, unsubscribe or reset health plan. Other classes associate with the patient class are address, health plan, usage and an appointment..

Doctors and the secretary of the clinic generalise from a staff class, where a Doctor has an attribute of role which is from a type enum (either doctor or hygienist). The secretary and doctors class contain several different methods but both of them contain the method to view schedule.

Schedule class is associated with the doctor and secretary class. This class gets all types of appointments, is associated with appointmenttype and appointment. Schedule doesn't contain any attributes, but contains methods that allow you to filter the appointment in numerous ways such as by doctor.

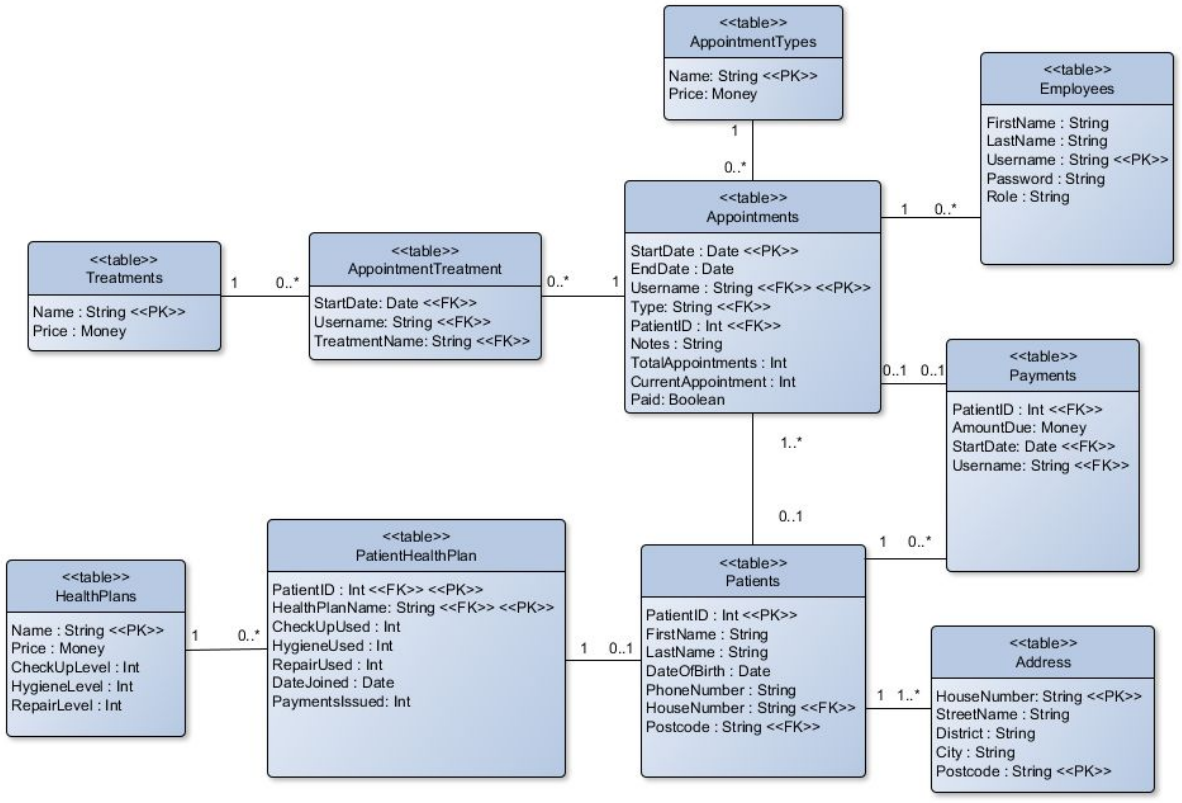
All classes have get methods, however our diagram only show classes where the methods are important to the class such as the schedule class.



Normalised Data Model Diagram

The figure below is our representation of a normalised data model, in which we tried to capture the relevant information from the scenario.

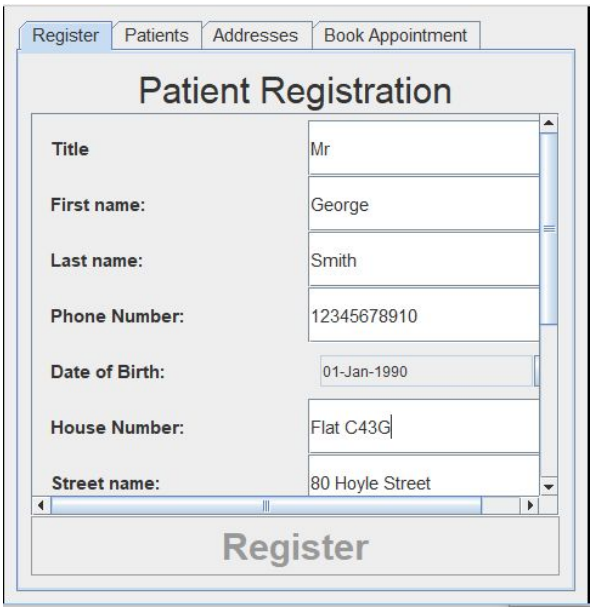
As you can see below we have decided to make two major tables, patient and appointments. These two table are associated with each other. All other tables are linked with these two tables in some way (directly associated or indirectly associated). Evidently this can be seen by the amount of foreign keys in the appointments class. All tables have a primary key to uniquely identify its data, except for appointment treatment as all it's attributes are foreign keys.



Query Processing

Registering a new patient and then showing that the new patient exists in the DB.

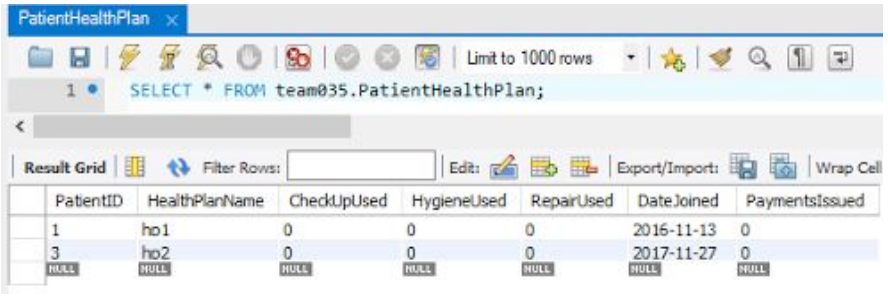
```
mysql> SELECT * FROM Patients;
+-----------+-------+-----------+----------+-------------+-------------+-------------+----------+
| PatientID | Title | FirstName | LastName | DateOfBirth | PhoneNumber | HouseNumber | Postcode |
+-----------+-------+-----------+----------+-------------+-------------+-------------+----------+
|         1 | Mr.   | la        | la       | 1997-02-12  | 02748593488 | Flat C43F   | S3 7LG   |
|         2 | Dr.   | as        | as       | 1993-06-13  | 08648148426 | Apt. E42F   | S3 7LS   |
+-----------+-------+-----------+----------+-------------+-------------+-------------+----------+
2 rows in set (0.00 sec)
```



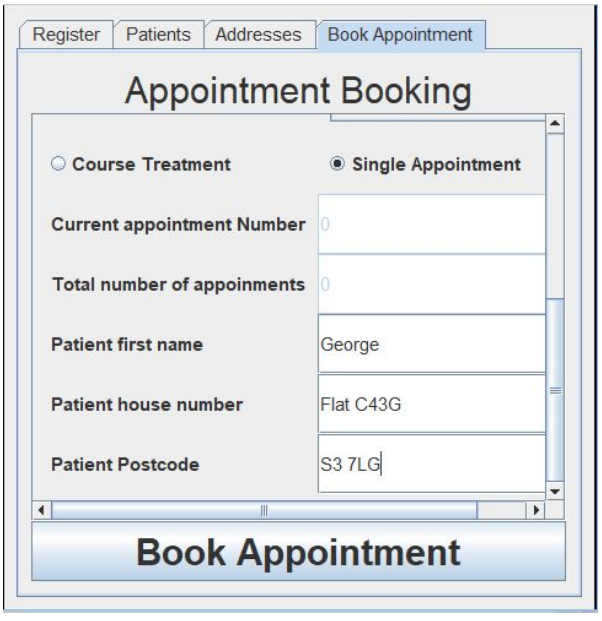

```
mysql> SELECT * FROM Patients;
+-----------+-------+-----------+----------+-------------+-------------+-------------+----------+
| PatientID | Title | FirstName | LastName | DateOfBirth | PhoneNumber | HouseNumber | Postcode |
+-----------+-------+-----------+----------+-------------+-------------+-------------+----------+
|         1 | Mr.   | la        | la       | 1997-02-12  | 02748593488 | Flat C43F   | S3 7LG   |
|         2 | Dr.   | as        | as       | 1993-06-13  | 08648148426 | Apt. E42F   | S3 7LS   |
|         3 | Mr    | George    | Smith    | 1990-01-01  | 12345678910 | Flat C43G   | S3 7LG   |
+-----------+-------+-----------+----------+-------------+-------------+-------------+----------+
3 rows in set (0.00 sec)
```

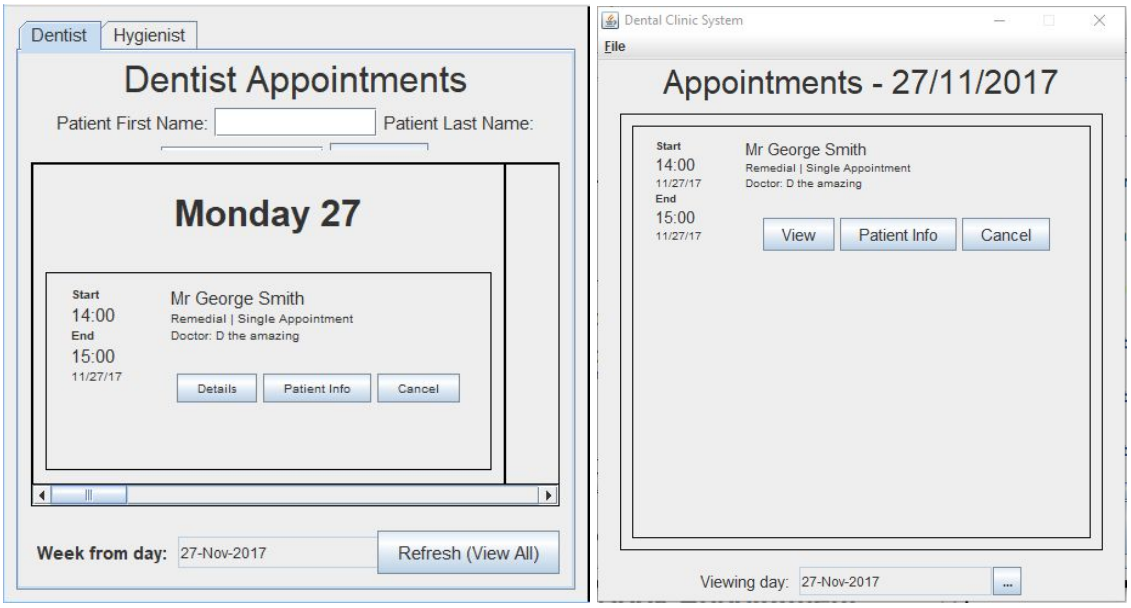
Subscribing a patient to a healthcare plan and then showing that the patient is linked with the relevant plan in the DB, through a new record of relevant treatment-credits for the plan, for the current year

```
mysql> SELECT * FROM PatientHealthPlan;
+-----------+----------------+-------------+-------------+------------+------------+----------------+
| PatientID | HealthPlanName | CheckUpUsed | HygieneUsed | RepairUsed | DateJoined | PaymentsIssued |
+-----------+----------------+-------------+-------------+------------+------------+----------------+
|         1 | hp1            |           0 |           0 |          0 | 2016-11-13 |              0 |
+-----------+----------------+-------------+-------------+------------+------------+----------------+
1 row in set (0.01 sec)
```



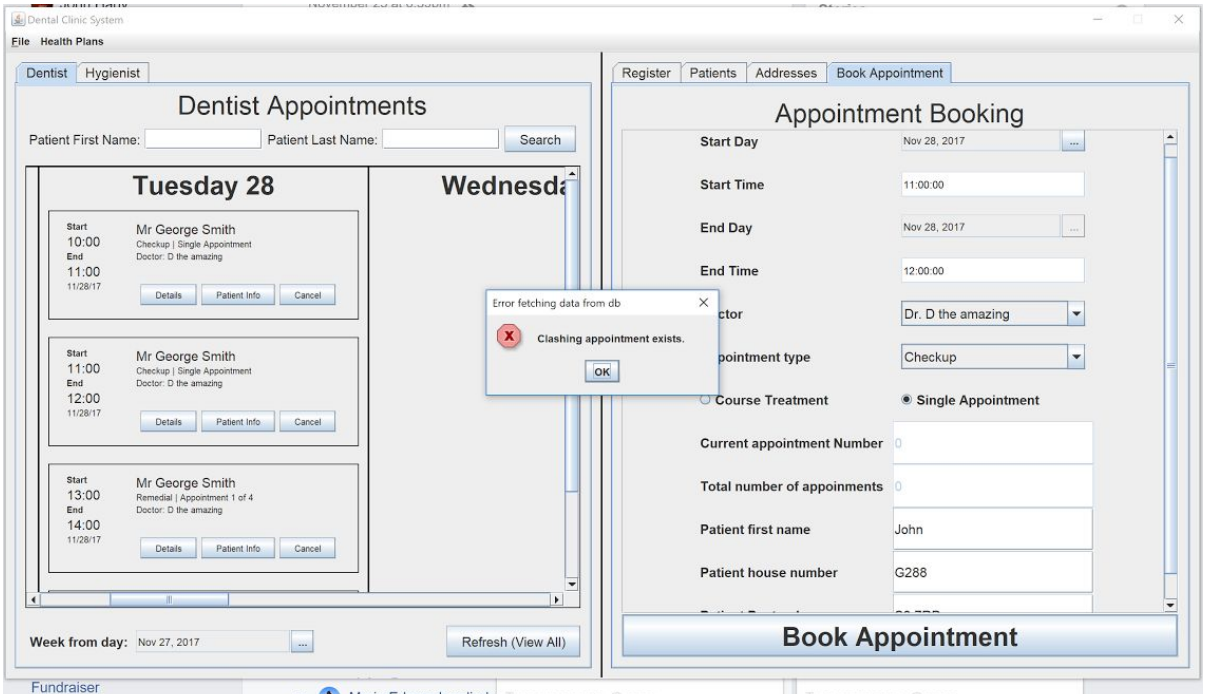
Creating an appointment for a patient to see the dentist and showing that this appointment appears in the receptionist's week-to-view calendar for the dentist, and also appears in the dentist's appointments for that day



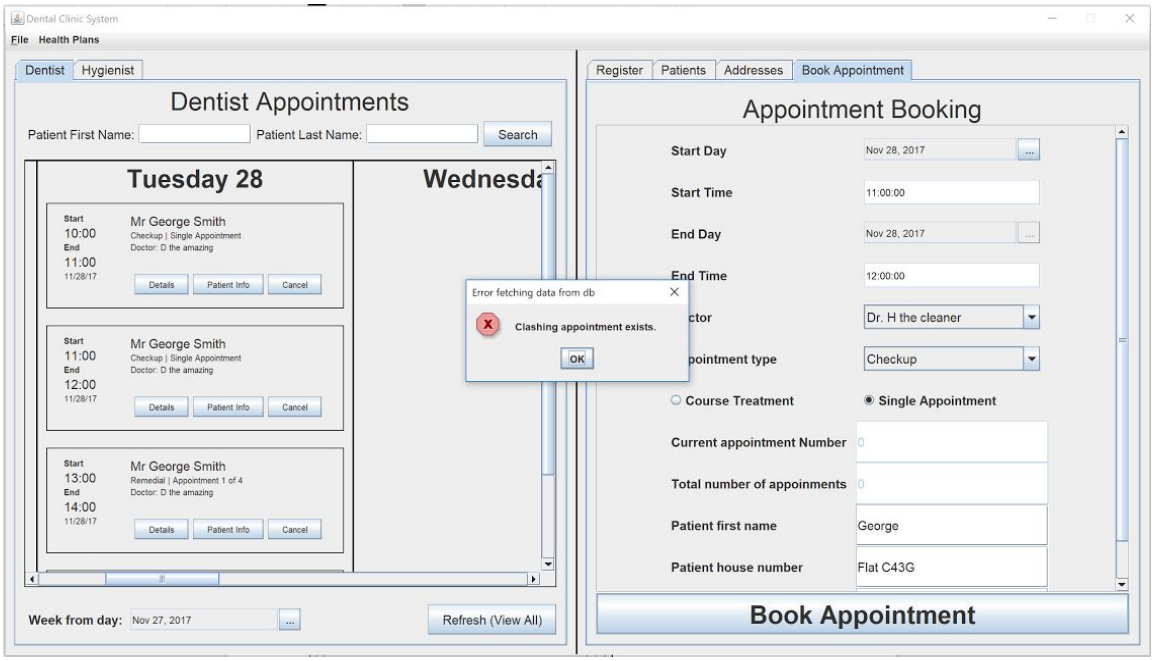


Attempting to create two appointments for a patient that are refused, either because the patient or the partner already have appointments at this time

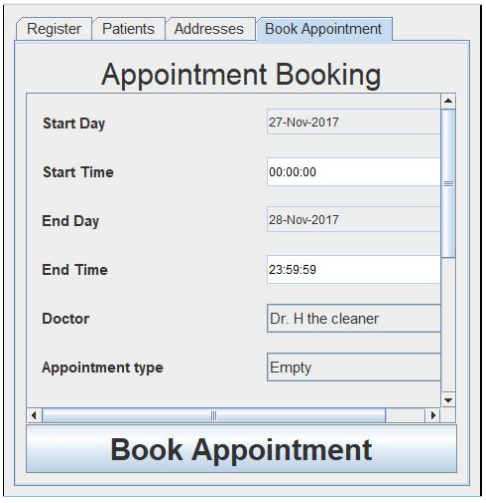
1) *Doctor has another appointment at the same time*



2) Patient has an appointment with the other doctor at the same time

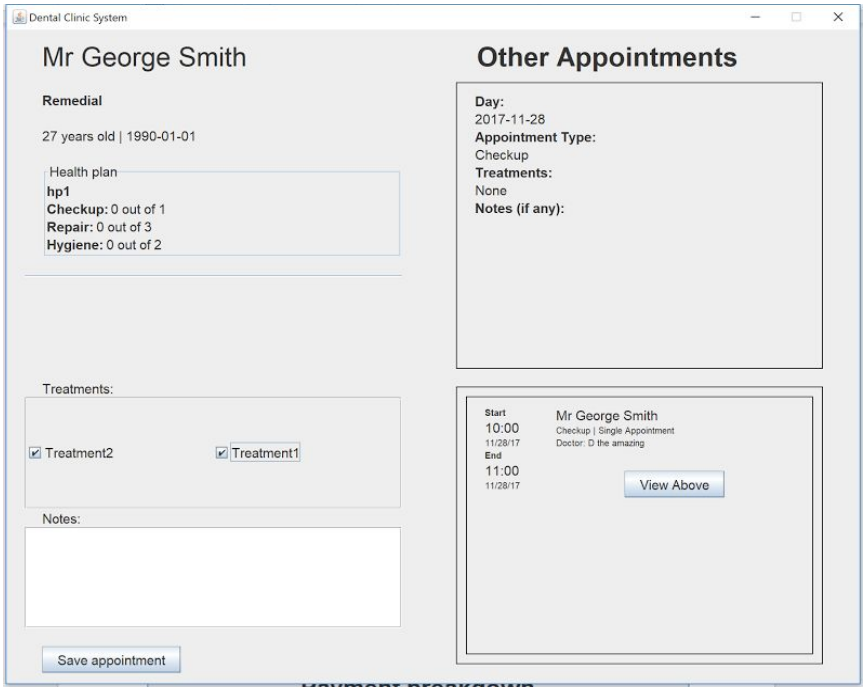


Booking two days holiday for the hygienist and then showing that blank appointments fill the relevant two days on the hygienist's week-to-view calendar

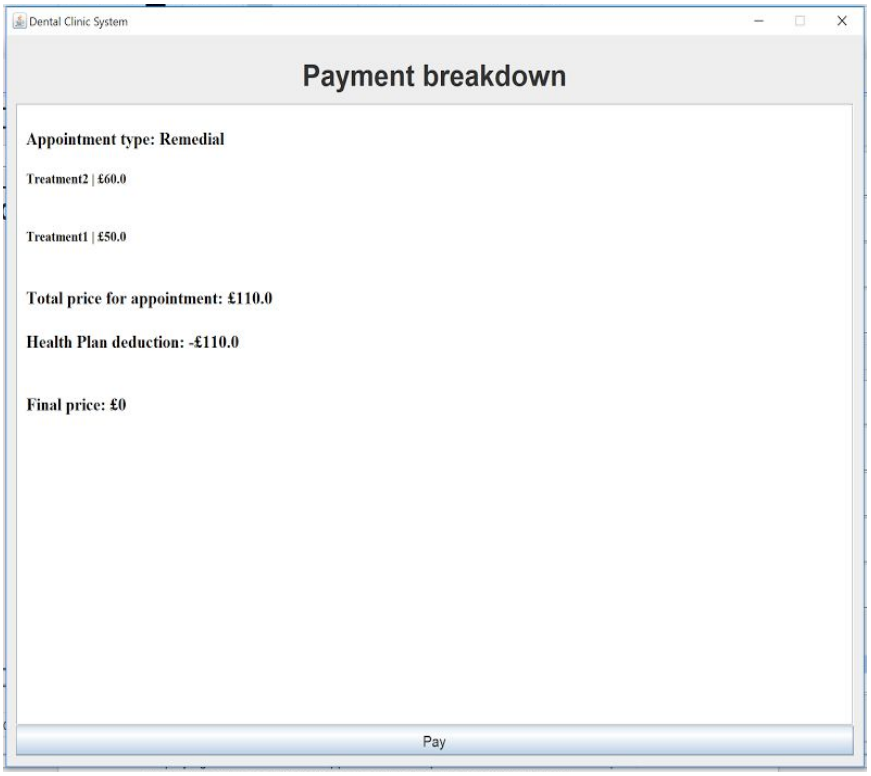


Recording two treatments given by the dentist to a patient and then showing that these have been added to the bill for the appointment

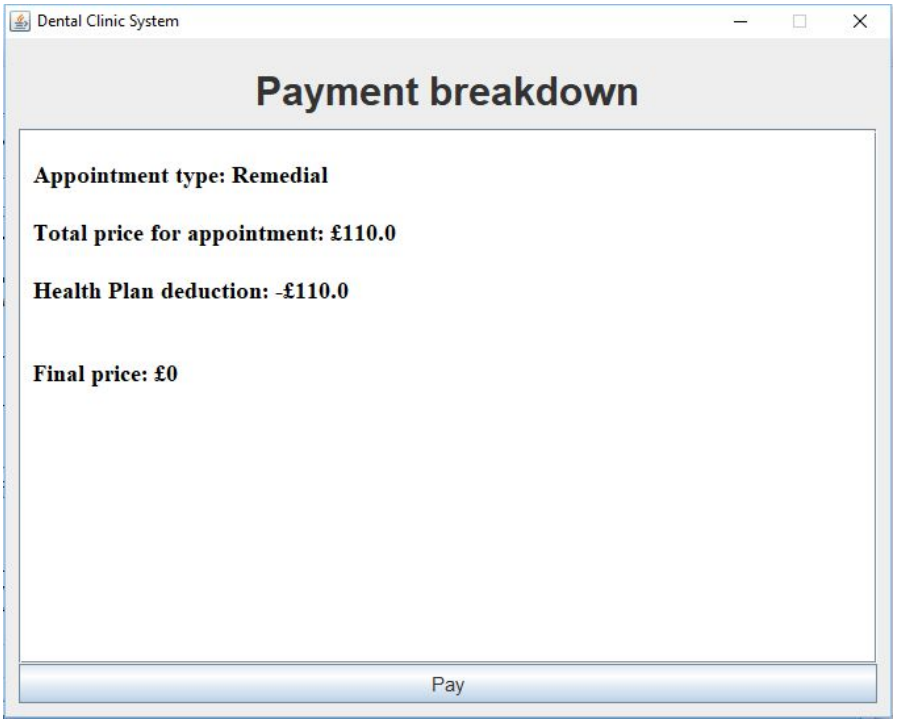
1) Doctor adding treatments



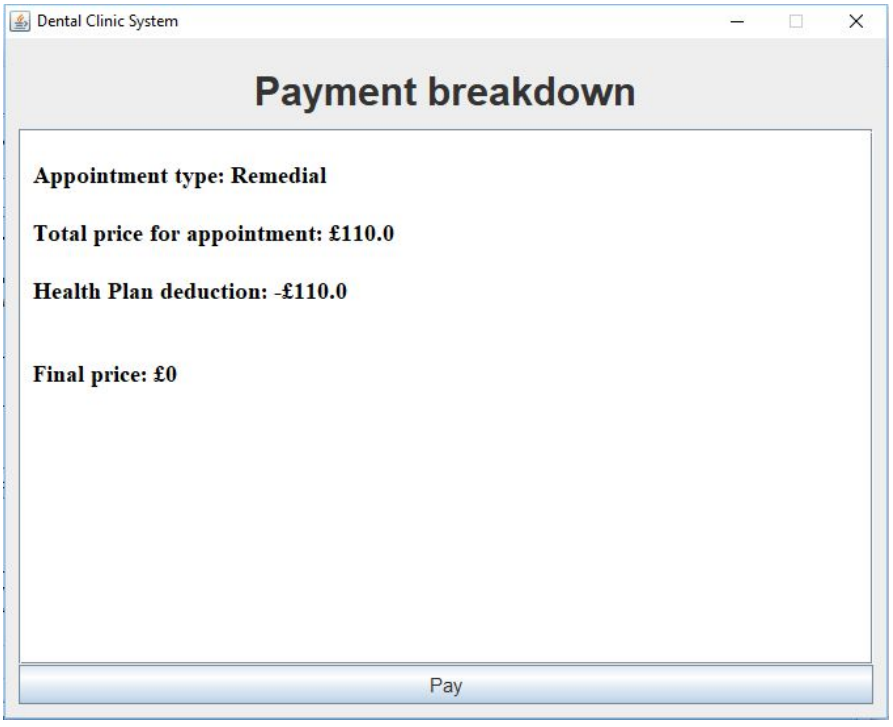
2) Secretary sees treatments in payment breakdown when patient is paying



Displaying the total cost of an appointment for a patient who is on a healthcare plan, showing the total cost of treatments and the amount owed by the patient



Processing a payment by a patient who is on a healthcare plan, showing how their treatment-credits for that year are adjusted, and that they now owe nothing.



Evaluation

On the whole, we believe we have performed very well as all requirements of the project have been fulfilled and even exceeded in many cases. Nonetheless, we felt there was room for improvement in our organisation, communication and efficiency.

Throughout the whole project we were well organised, every meeting was attended by us all with the exception of one or two meetings. However towards the end of the project we had discussed that the distribution of work could have been distributed more evenly, as some of us thought we had too much work, this leads us to believe our organisation could have been better. The failure here comes from the fact that certain aspects of the system were harder/easier than expected. Knowledge of these things comes with experience, so it's not too surprising that this happened.

What's good though is that we were all willing to help each other and discuss ways of improving our system so this eased the burden of some. Our frequent meetings, in addition to our group chat that we set up, helped in this regard.

Sometimes in-house deadlines weren't met by us all, but this is to be expected since workload varies throughout the semester.

On the whole we believe we were efficient as the software was finished on time and to a high standard.

Signed effort Deceleration

Team Member	Work Contributed	Effort
Andreas Constantinou	● Contributed to diagrams ● Created schedule class ● Contributed to appointment class ● Contributed to model tests ● Contributed to report	85%
Arthur Granacher	● Contributed to diagrams ● Created appointment class ● Created appointment type Enum ● Created DBQueries class ● Create Role type Enum ● Created Doctors class ● Created Secretary class ● Created Staff class ● Created model tests ● Contributed to report	100%
John Ayad	● Main contributor to diagrams ● Creator of Interface package and its class (Only contributor) ● Creator of view package and its classes (Only contributor) ● Created and Initialised Database package (Only contributor) ● Created setupwizard package (Only contributor) ● Contributed health plan class ● Implemented design of all of the application (views and controllers) (only contributor) ● Contributed to the models ● Creator of utils package and its classes (Only contributor) ● Contributed to report	115%
Nuraldeen Magid	● Contributed to diagrams ● Contributed to model tests ● Created patients class ● Created address class ● Created usage class ● Created health plan class ● Contributed to report	100%

Signature

Nuraldeen Magid Arthur Granacher John Ayad Andreas Constantinou