**Implementation and Testing**

This chapter builds upon what was discussed previously in the design chapter. It details key areas relating the development of the system throughout the various sprints that were undertaken. During each sprint, the developer undertook unit testing to ensure that the required functionality was successfully delivered. As has been mentioned previously, the client was actively involved in the sprints, and was presented with regular demonstrations of system functionality during each sprint.

This chapter will be structured based on the order in which the sprints were undertaken, with each of these sprints being undertaken in order of importance.

**Users**

The system will be managing sensitive documents in the form of exam documentation, with access to these documents and the system needing to be restricted to relevant individuals. Should an unauthorised user gain access to the system, it would undermine the integrity of the School of Computing’s assessments and raise questions about the authenticity of qualifications achieved by alumni. Furthermore, there are a variety of distinct roles involved in the examination process, with each user group possessing differing needs for the system. By employing user accounts the system is better placed to authenticate users and tailor their experience of the system to their role.

Within Django there are three commonly used techniques relating to managing user accounts, these being:

* Base user model – This is the standard user model within Django. It contains three key fields, these being username, password and email address and a few additional fields such as superuser, active, date joined, etc. It offers full integrations with Django’s authentication system out the box, such as managing signing in and password resets. However, it has the disadvantage that the user model cannot be adjusted, for example adding additional fields.
* Abstract user model – Essentially this user model maintains all the functionality in relation to authentication as the base user model. It has a significant advantage in that it can be customised to include additional fields, even after the first database migration. As a result of this added customisation, it is recommended to adopt an abstract user model even if the base user models functionality is sufficient (<https://docs.djangoproject.com/en/2.2/topics/auth/customizing/>).
* Abstract base user – This implementation offers the greatest degree of flexibility of the three main user models. Its main advantage is it offers the advantages of the abstract user model, while also allowing the altering of the underlying user authentication functionality. For example, if there was a desire to use an email address as opposed to a username for logging in, this would be achieved using an abstract base user. However, because of this increased flexibility, there is often an added development burden of having to modify the underlying methods of this model, such as the base user manager.

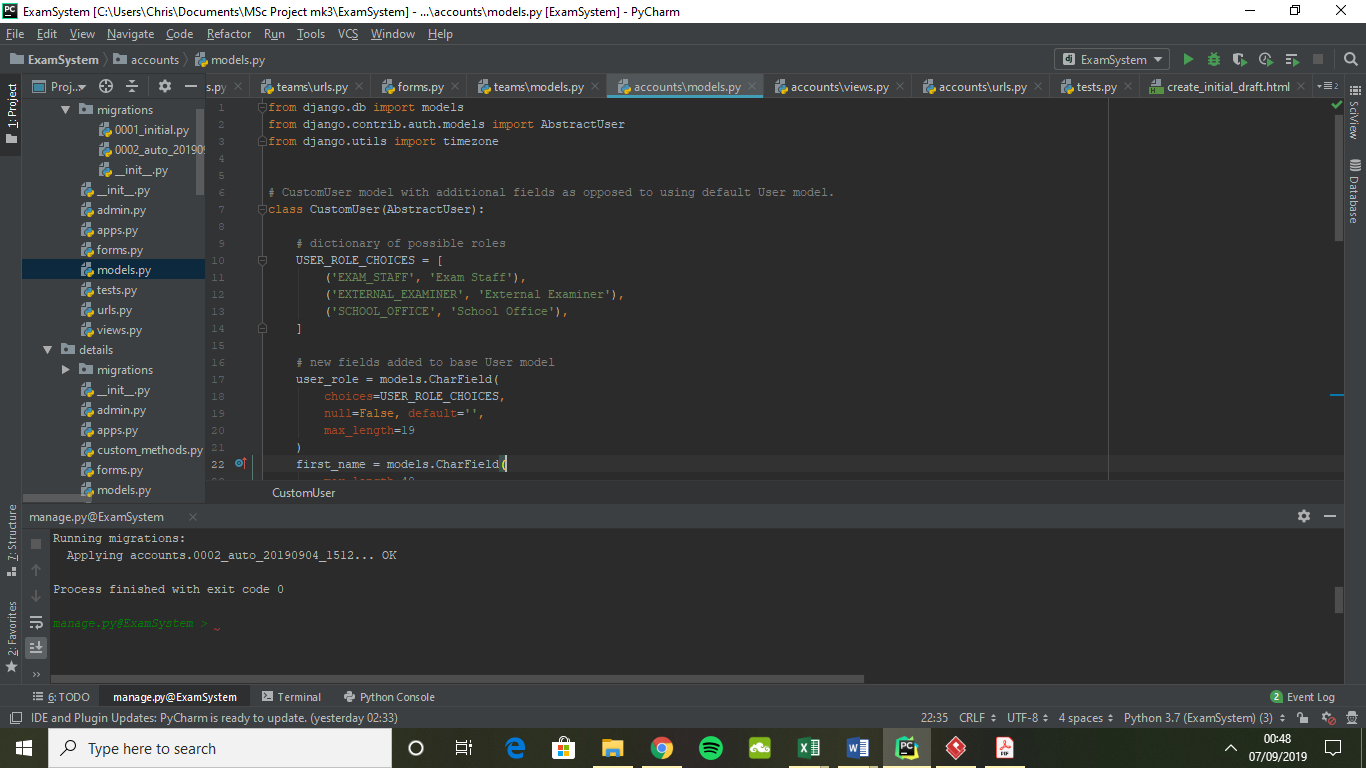
For the purposes of the project each of these approaches were considered. As has been stated previously, the security of the system and by continuation its user authentication are of the utmost importance. There was a consideration of using staff emails as the user login and as a result adopt an abstract base user model approach. The benefit of this is staff would not need to remember a username. However, this presents a disadvantage as these email addresses are publicly available on the university’s website (<https://www.dundee.ac.uk/computing/people/>). Subsequently if staff emails were used, half of the authentication details would be publicly available, making it easier for unauthorised users to gain access.

In terms of adopting the base user model, as mentioned even the developers of Django recommend against this even when it contains all the functionality required. As a result, the abstract user approach was adopted, which will also allow future extendibility of the user model, should it be required.

The developer proposed that the gathering of additional user specific information beyond simply a username, password and email address would be beneficial. Specifically, these include:

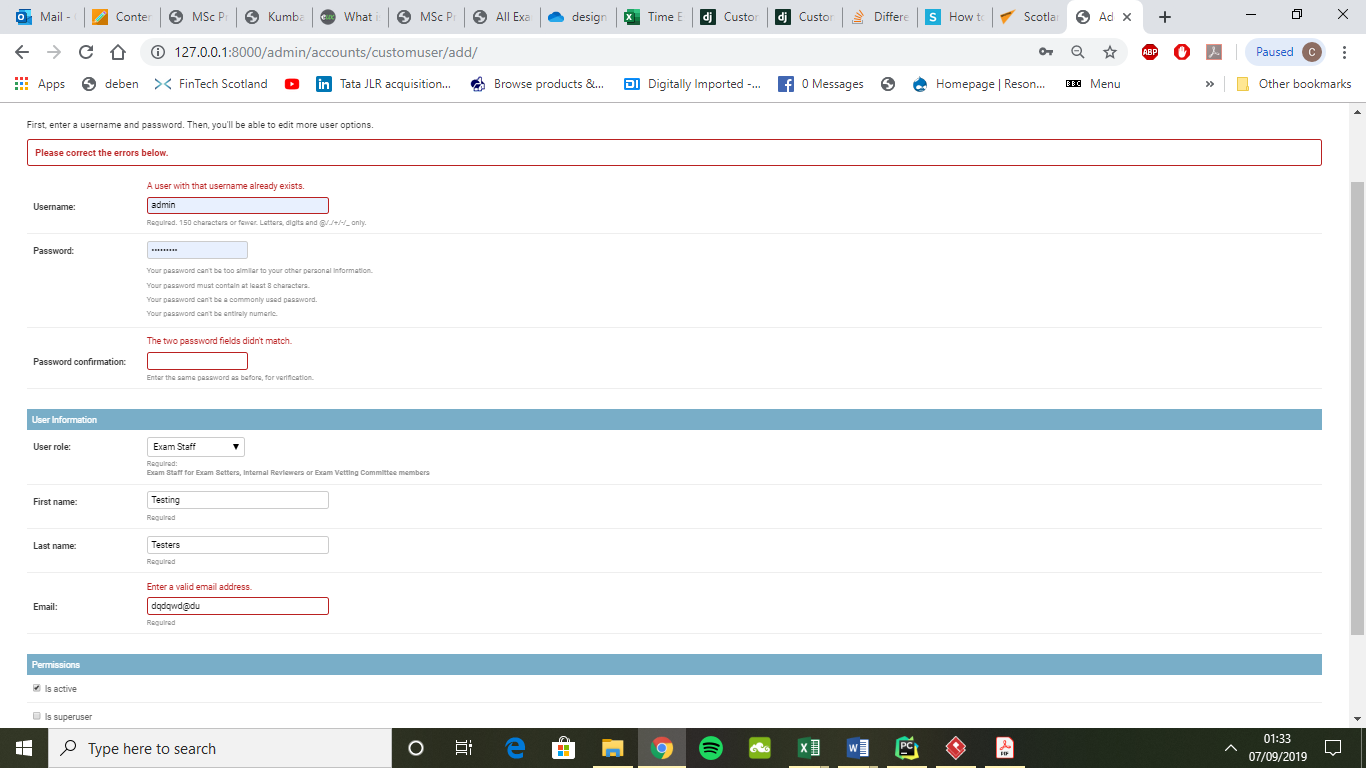
* First and last name – These would be useful in the system as many of the staff using EMS will already know their colleagues’ names. By collecting users’ actual names, the system can aid user identification between users.
* User role – This would be used to assign the user to their specific roles related to the exam review process.

In terms of adding these fields to the system, there are two possible approaches. Either another table could be created which links to the user model through a one-to-one foreign key relationship, or they could be added to the abstract user model. The case of moving the user role to another table is an interesting one as it would aid normalisation of the database. However, the developer did not believe doing so at the expense of more complex querying was justified. Instead first and last name were added to the user table, with user role also included as mandatory choice field **(SEE DIAGRAM)**



Exam staff encompasses exam setters, internal reviewers and the vetting committee. This was justified as the internal exam staff are often involved in the setting of multiple exams and regularly occupy these different roles. As a result, by grouping these roles together a user can perform these tasks from a single account. External examiners and the school office are for their corresponding roles in the presently adopted system. The local examination officer also has the role of internal exam staff, so would have ‘Exam Staff’ as their role. However, they perform additional administrative responsibilities in the existing system. In EMS the local examination officer is set as a superuser, which grants them some additional functionality on the main site and access to the administration site.

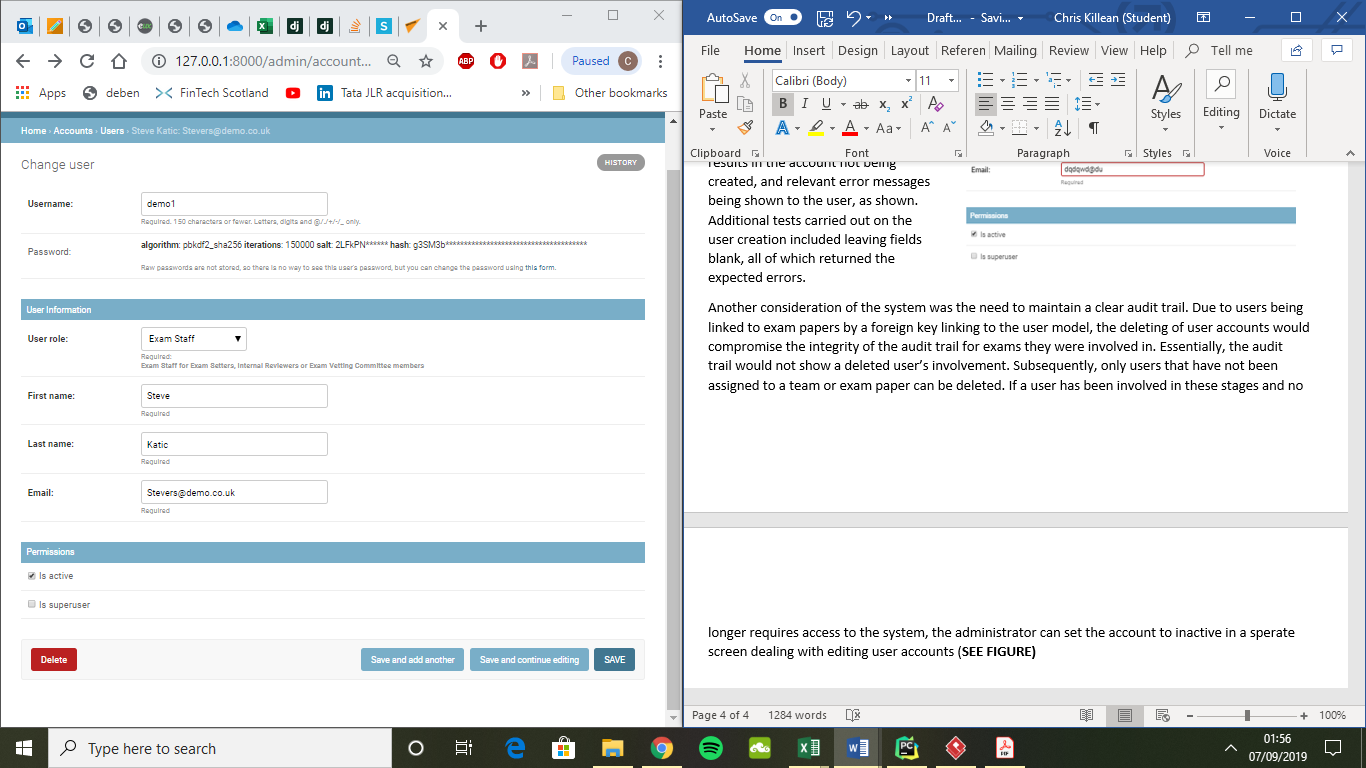
**Administration of User Accounts**

Consideration was given as to whether users would be able to register their own accounts, or if it would be carried out by an administrator. While users creating their own accounts would spread the workload, it could lead to unwelcome users registering. If accounts are created by an administrator there would be an additional start-up cost to the system in terms of time, but this burden would be reduced in future years as most accounts would already have been created. As such a restricted registration approach was adopted whereby the local examination officer would create accounts using the administration panelas shown in (**INSERT FIGURE).**

This form takes all the key information needed for a user account during registration. Additionally, when a form is submitted it validates the information entered. This includes ensuring:

* Username is not already in use
* Passwords match each other, are not commonly used passwords or similar to the username.
* The email address is not already in use in the system, as well as ensuring it is in a valid format.

Failure to meet the criteria of a field results in the account not being created, and relevant error messages being shown to the user, as shown. Additional tests carried out on the user creation included leaving fields blank, all of which returned the expected errors.

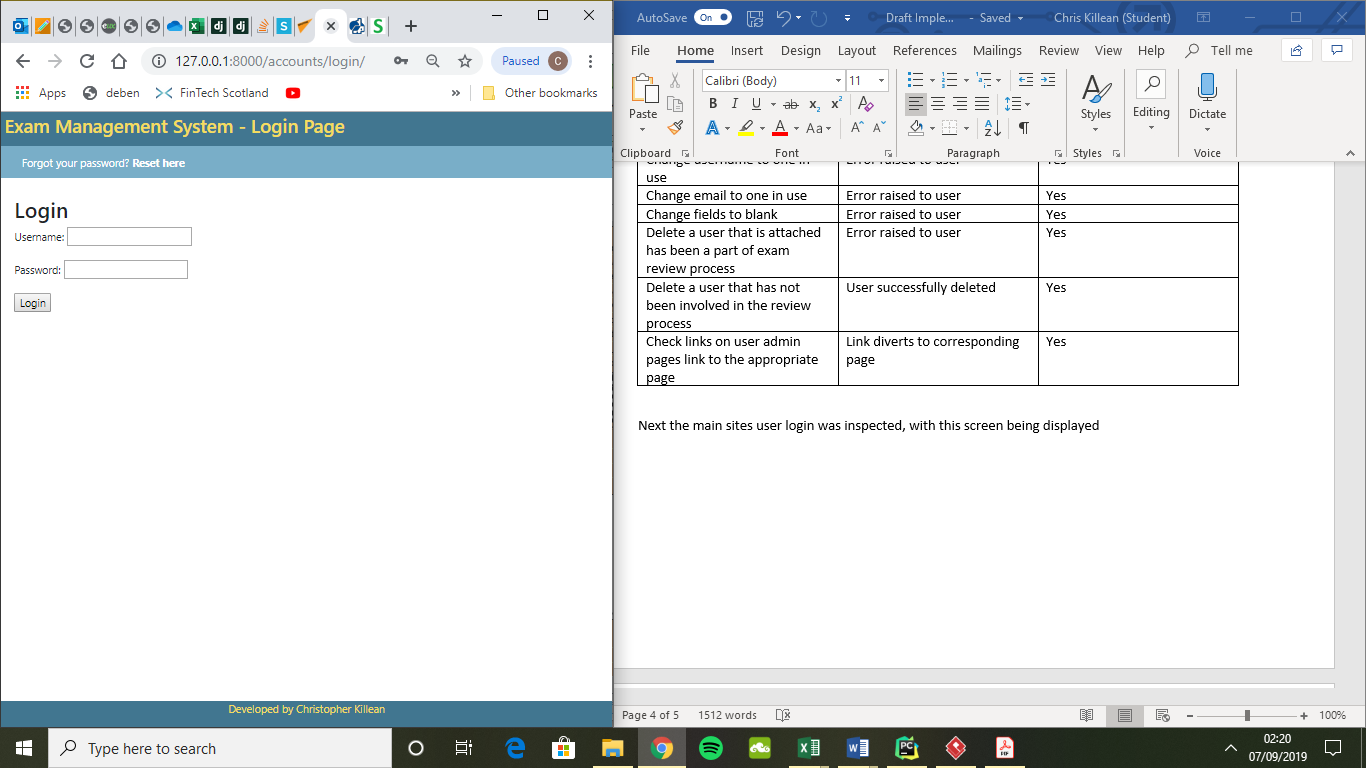
Another consideration of the system was the need to maintain a clear audit trail. Due to users being linked to exam papers by a foreign key linking to the user model, the deleting of user accounts would compromise the integrity of the audit trail for exams they were involved in. Essentially, the audit trail would not show a deleted user’s involvement. Subsequently, only users that have not been assigned to a team or exam paper can be deleted. If a user has been involved in these stages and no longer requires access to the system, the administrator can set the account to inactive in a sperate screen dealing with editing user accounts (**SEE FIGURE).**

Coincidently, this screen also shows an important security feature of how the system protects passwords. Essentially the password is encrypted on two levels, first through a hashing algorithm and secondly through a user unique salt algorithm. This means that even if someone was able to get access to the database and decrypt one user’s password, it would be unlikely they would immediately be able to decrypt all other user passwords.

**Testing**

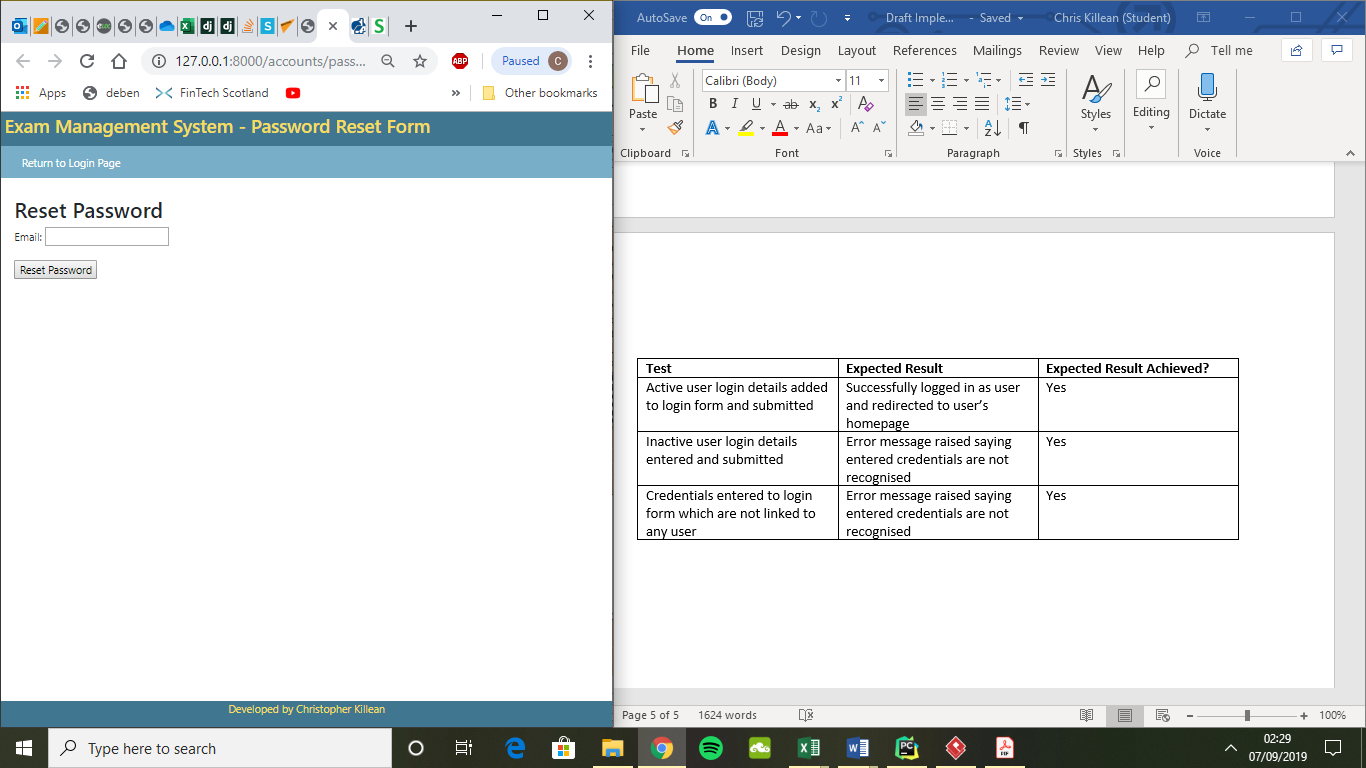
This section will cover the testing undertaken for the user accounts of the system. These are in addition to the tests covered previously for the creation of users. The first area looked at is the admin sites user administration functionality. As can be seen from the tests conducted in **FIGURE XXXXX** all expected results were achieved.

|  |  |  |
| --- | --- | --- |
| **Test** | **Expected Result** | **Expected Result Achieved?** |
| Change username to one in use | Error raised to user | Yes |
| Change email to one in use | Error raised to user | Yes |
| Change fields to blank | Error raised to user | Yes |
| Delete a user that is attached has been a part of exam review process | Error raised to user | Yes |
| Delete a user that has not been involved in the review process | User successfully deleted | Yes |
| Check links on user admin pages link to the appropriate page | Link diverts to corresponding page | Yes |

Next the main sites user login was inspected, with this screen being displayed in **FIGURE XXXXX**.

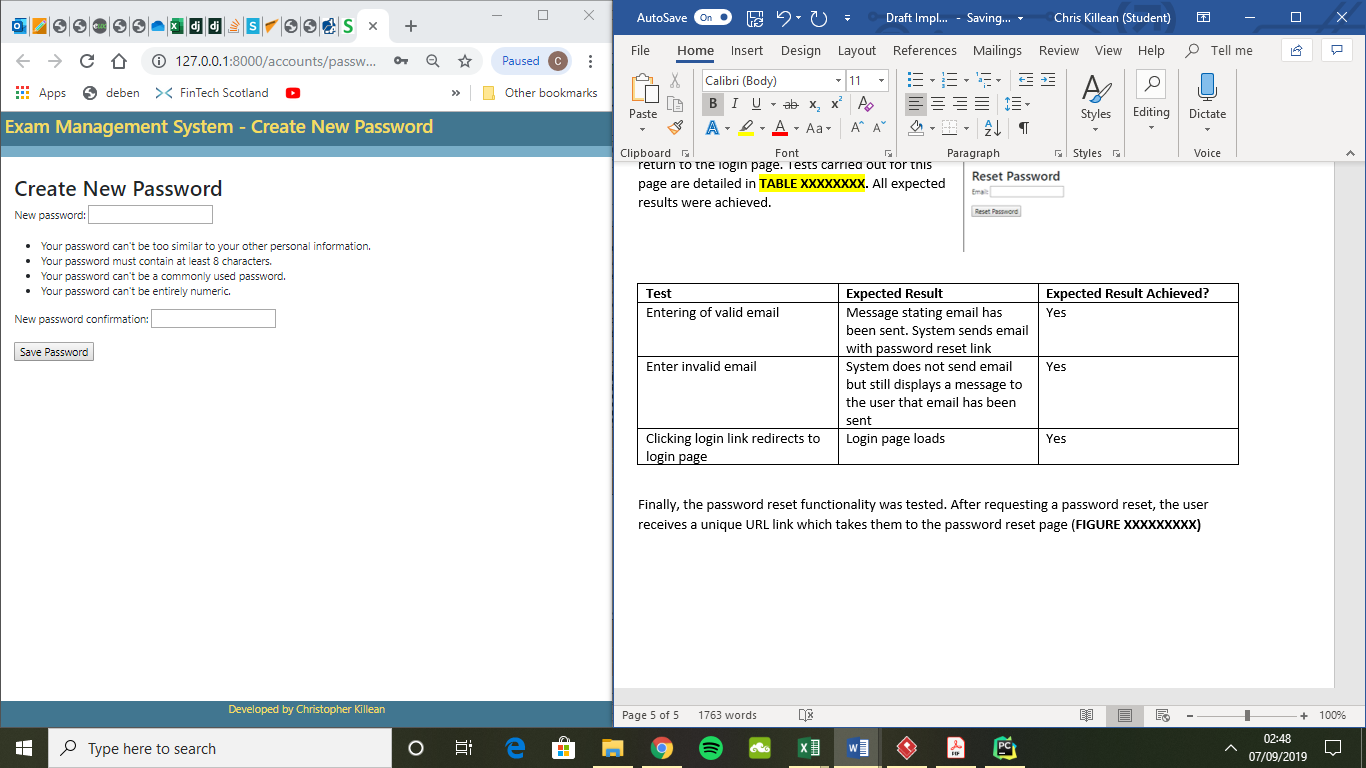
It should be noted this figure does not show footer which appears on the login page. From this page the user can login or reset their password. Tests on the login functionality are tested in **TABLE XXXXX**. These tests provided expected results.

|  |  |  |
| --- | --- | --- |
| **Test** | **Expected Result** | **Expected Result Achieved?** |
| Active user login details added to login form and submitted | Successfully logged in as user and redirected to user’s homepage | Yes |
| Inactive user login details entered and submitted | Error message raised saying entered credentials are not recognised | Yes |
| Credentials entered to login form which are not linked to any user | Error message raised saying entered credentials are not recognised | Yes |
| Clicking Forgot password redirects to password reset page | Password reset page displayed | Yes |

The password reset page contains a field for the entering of an email address, as well as link to return to the login page. Tests carried out for this page are detailed in **TABLE XXXXXXXX.**

The results are detailed in **TABLE XXXXXXX**, with all expected results being achieved.

|  |  |  |
| --- | --- | --- |
| **Test** | **Expected Result** | **Expected Result Achieved?** |
| Entering of valid email | Message stating email has been sent. System sends email with password reset link | Yes |
| Enter invalid email | System does not send email but still displays a message to the user that email has been sent | Yes |
| Clicking login link redirects to login page | Login page loads | Yes |

Finally, the password reset functionality was tested. After requesting a password reset, the user receives a unique URL link to their email address which takes them to the password reset page (**FIGURE XXXXXXXXX)**.

All tests for this page were successful, with the tests carried out outlined in **TABLE XXXXXXX**.

|  |  |  |
| --- | --- | --- |
| **Test** | **Expected Result** | **Expected Result Achieved?** |
| Passwords entered match and are not similar to username | Password changed, redirect to login page | Yes |
| After successful password reset - On login page, attempt to login with new password | Successfully login and redirect to homepage | Yes |
| After successful password reset – On login page, attempt to login with old password | Raise error, credentials not recognised | Yes |
| Passwords entered match but are similar to username | Raise errors to user saying passwords are too similar to username | Yes |
| Enter non matching passwords | Raise error that passwords don’t match | Yes |
| Leave fields blank and submit | Raise error that password is too short | Yes |

**Client Demonstration**

After the functionality of user accounts were created it was demonstrated to the client. The client was happy with what had been implemented.

**Teams**

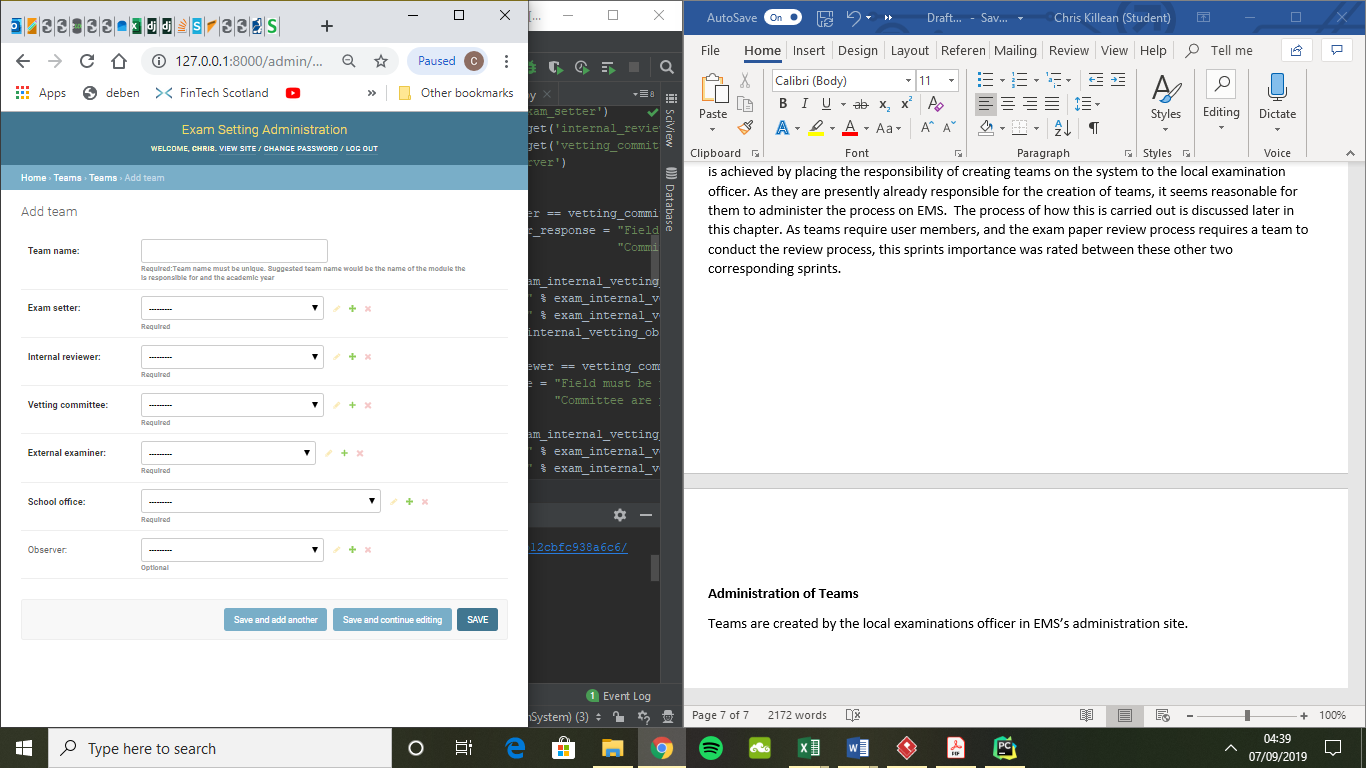
Each exam paper in the presently adopted system has a team, with each member performing a specific role of the review process for an exam paper. Each semester, a list of review teams for each module is distributed amongst those involved in the exam setting process. Additionally, as there is a relatively limited number of individuals involved in the exam setting process within the school, there is the possibility that the same team, with users occupying the same roles could exist across multiple modules exams. Furthermore, a module’s exam review team may not change year on year.

EMS sought to introduce the functionality of teams, where the team could be reused over multiple years (in the event team makeup hasn’t changed) and across multiple modules. This functionality also sought to do away with the need to distribute a teams list, as is common practice presently. This is achieved by placing the responsibility of creating teams on the system to the local examination officer. As they are presently already responsible for the creation of review teams in the presently adopted system, it seems reasonable for them to administer the process on EMS. As teams require user members, and the exam paper review process requires a team to conduct the review process, this sprints importance was rated between these other two corresponding sprints.

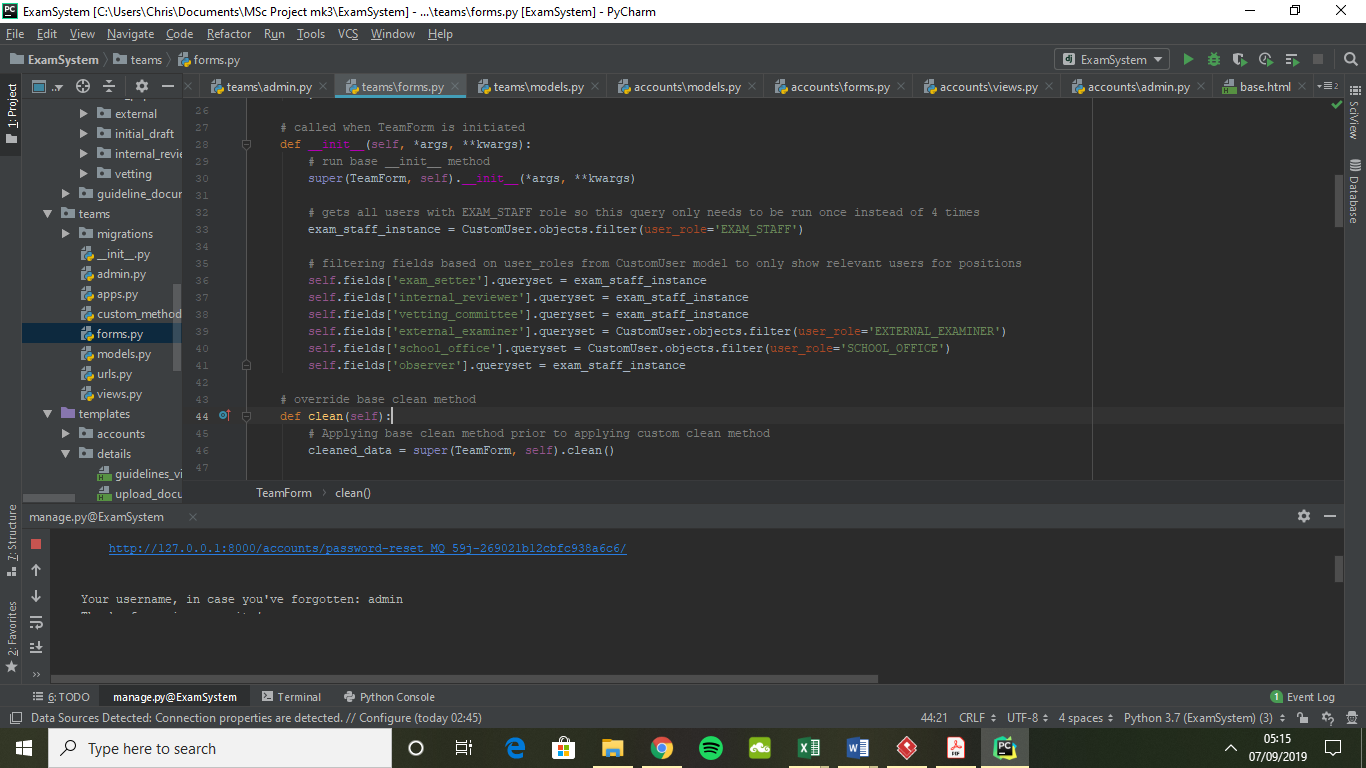
**Administration of Teams**

This section details the implementation and testing of functionality for adding and editing teams.

**Creating Teams**

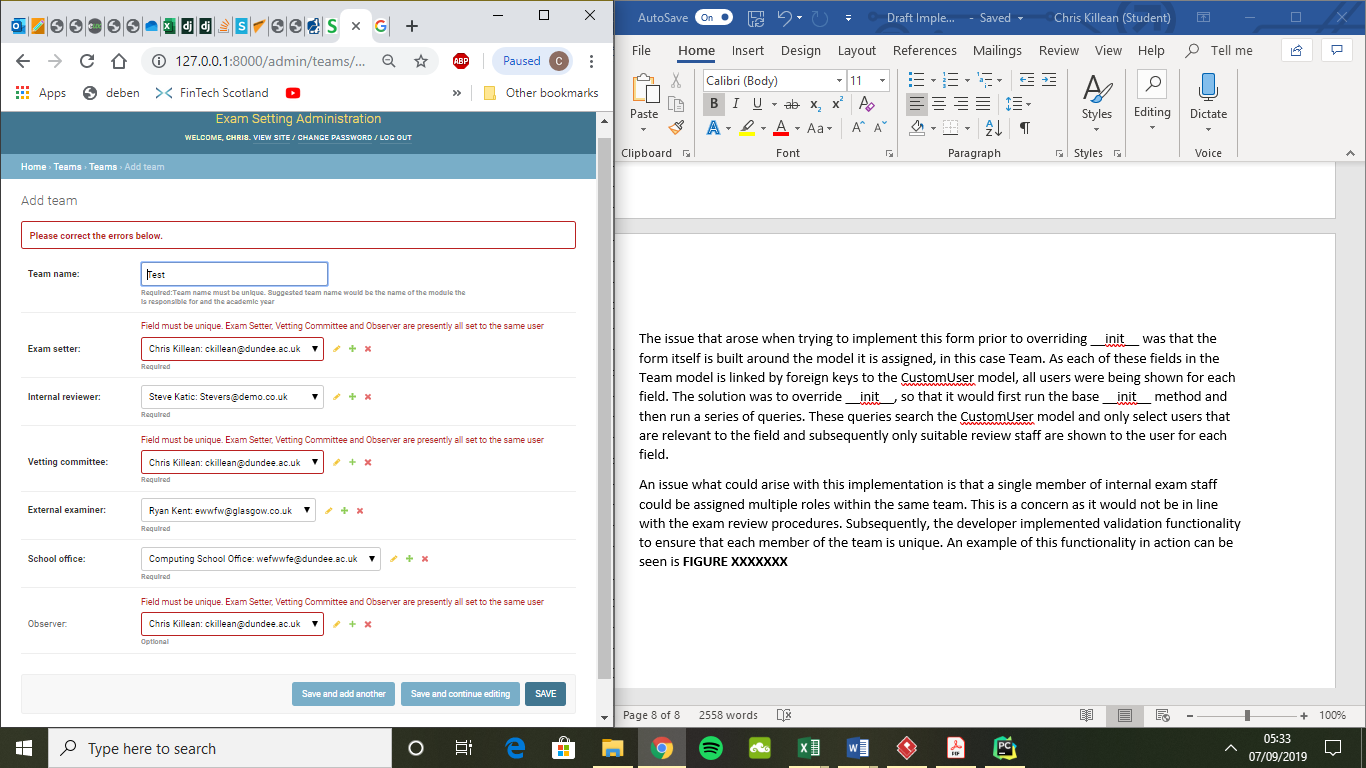
Teams are created by the local examinations officer in EMS’s administration site. The page for creating teams can be seen in **FIGURE XXXXXX**.

All fields are required, except for ‘Observer’ which is an optional field. Discussions with the client revealed that some modules will have an additional individual who requires oversight of the exam review process, such as in instances where there are two module leaders, and as such this is the purpose of the observer team member. The team name is used so that users of the system can distinguish between teams. Other fields represent distinct roles for the review process. Each of these dropdown menus only offers selection options of users who have the corresponding role profile selected in their user profile. To achieve this functionality and only have the relevant users displayed for each dropdown menu field, the \_\_init\_\_ constructer method for the TeamForm class was overridden **(FIGURE REF)**. It is important to note that the developer made use of forms.ModelForm for the forms used in this project. This is a key advantage of Django development as it automatically adds basic functionality to the form which otherwise would normally be boilerplate code.



The issue that arose when trying to implement this form prior to overriding \_\_init\_\_ was that the form itself is built around the model it is assigned, in this case Team. As each of these fields in the Team model is linked by foreign keys to the CustomUser model, all users were being shown for each field. The solution was to override \_\_init\_\_, so that it would first run the base \_\_init\_\_ method and then run a series of queries. These queries search the CustomUser model and only select users that are relevant to the field and subsequently only suitable review staff are shown to the user for each field.

An issue what could arise with this implementation is that a single member of internal exam staff could be assigned multiple roles within the same team. This is a concern as it would not be in line with the exam review procedures. Subsequently, the developer implemented validation functionality to ensure that each member of the team is unique. An example of this functionality in action can be seen is **FIGURE XXXXXXX.**



This error checking is performed after the user attempts to save the new Team object instance. In the event validation fails, the new Team instance is not saved, and instead the user is presented with relevant error messages.

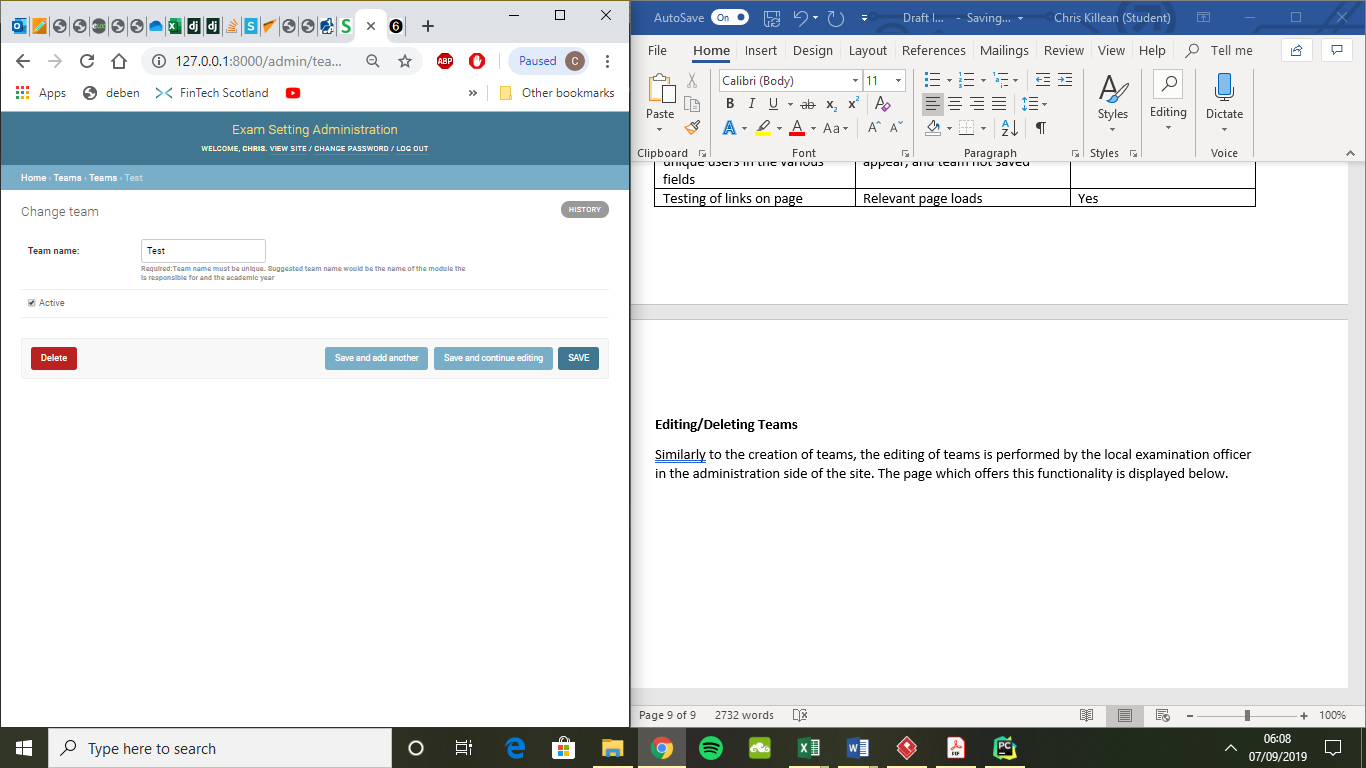
**Testing**

To test the functionality of creating teams, the following tests were performed (**TABLE REF**), all of which returned the desired results.

|  |  |  |
| --- | --- | --- |
| **Test** | **Expected Result** | **Expected Result Achieved?** |
| Enter all unique fields | New team saved, success message displayed | Yes |
| Enter unique team member fields and leave team name blank | Error message stating that team name is required, team not saved | Yes |
| Multiple attempts at submitting a form with non-unique users in the various fields | Return error messages for the fields where the repeated users appear, and team not saved | Yes |
| Testing of links on page | Relevant page loads | Yes |

**Editing/Deleting Teams**

Similarly, to the creation of teams, the editing of teams is performed by the local examination officer in the administration side of the site. The page which offers this functionality is displayed below.



As can be seen, not all fields are editable on this page. This is deliberate as foreign keys for exam papers will link to the users selected for each role of the responsible team object instance. As a result, if they were edited, these changes would be reflected on all associated exam papers, which would make their audit trail unreliable. Resultantly, a team can be set to inactive or have its team name changed. The team can only be deleted if it does not have any exam papers attached to the team. As with the team creation page, the team name cannot be blank and must be unique, there is appropriate validation in place to ensure this.

**Testing**

The following tests were carried out to ensure functionality performed as expected, which it did.

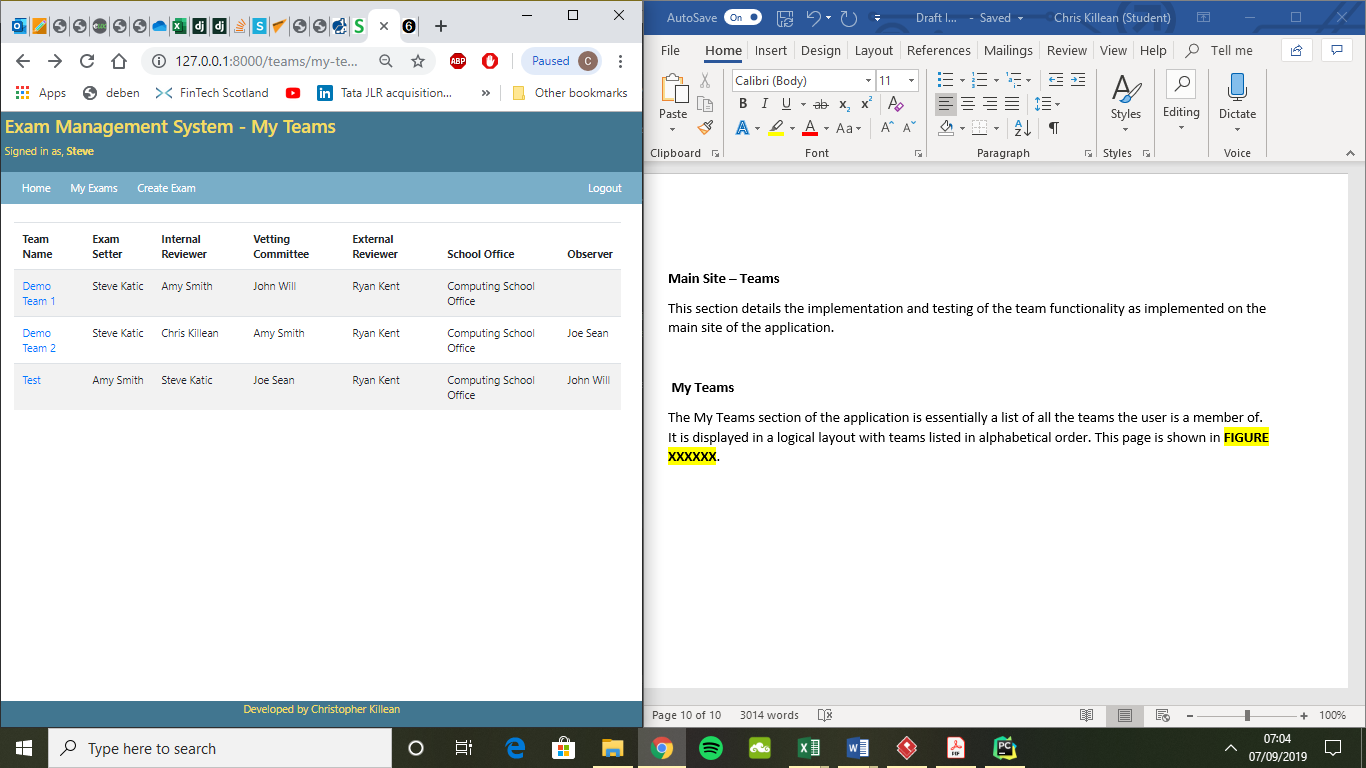
|  |  |  |
| --- | --- | --- |
| **Test** | **Expected Result** | **Expected Result Achieved?** |
| Name changed to unique team name | Team object saved with new team name | Yes |
| Name changed to non-unique or blank | Error message raised, indicating appropriate error for team name | Yes |
| Team that has no exam papers attached to it delete attempt | Team object successfully deleted | Yes |
| Team that has been involved in the setting of exam delete attempt | Error message raised stating that team cannot be deleted, with a list of related exam object instances listed | Yes |

**Main Site – Teams**

This section details the implementation and testing of the team functionality as implemented on the main site of the application

**My Teams**

The My Teams section of the application is essentially a list of all the teams the user is a member of. It is displayed in a logical layout with teams listed in alphabetical order. Team names serve as a clickable link to the corresponding teams homepage. This page is shown in **FIGURE XXXXXX**.



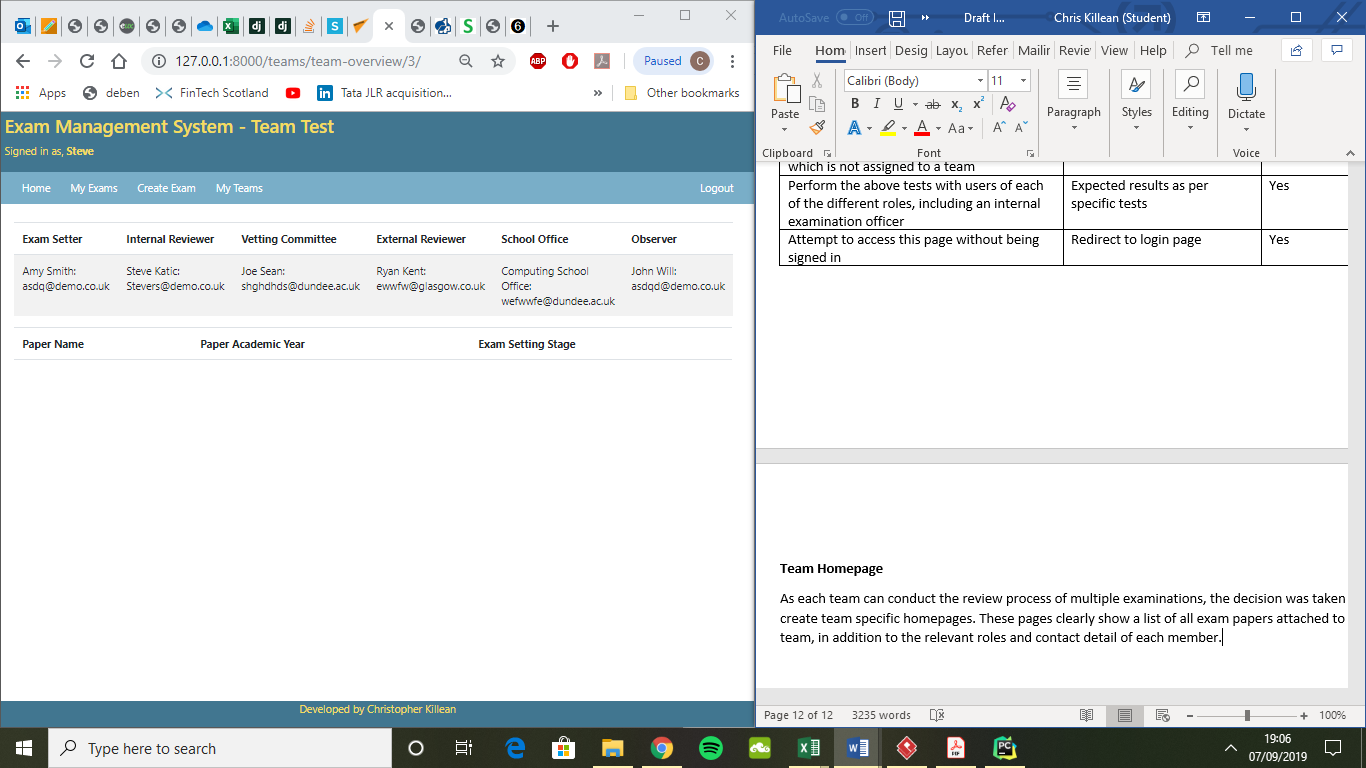
**Testing**

To validate the functionality of this page, the tests in the table below were carried out. It should be noted that when this page was initially created the exam functionality had not been implemented, and as such while the navbar links ‘My Exams’ and ‘Create Exam’ were present, they did not lead anywhere. At this stage of creation, the logic which ensured ‘Create Exam’ appeared only for internal exam staff had been implemented. These links were connected to the appropriate pages and tested once the exam review functionality had been added.

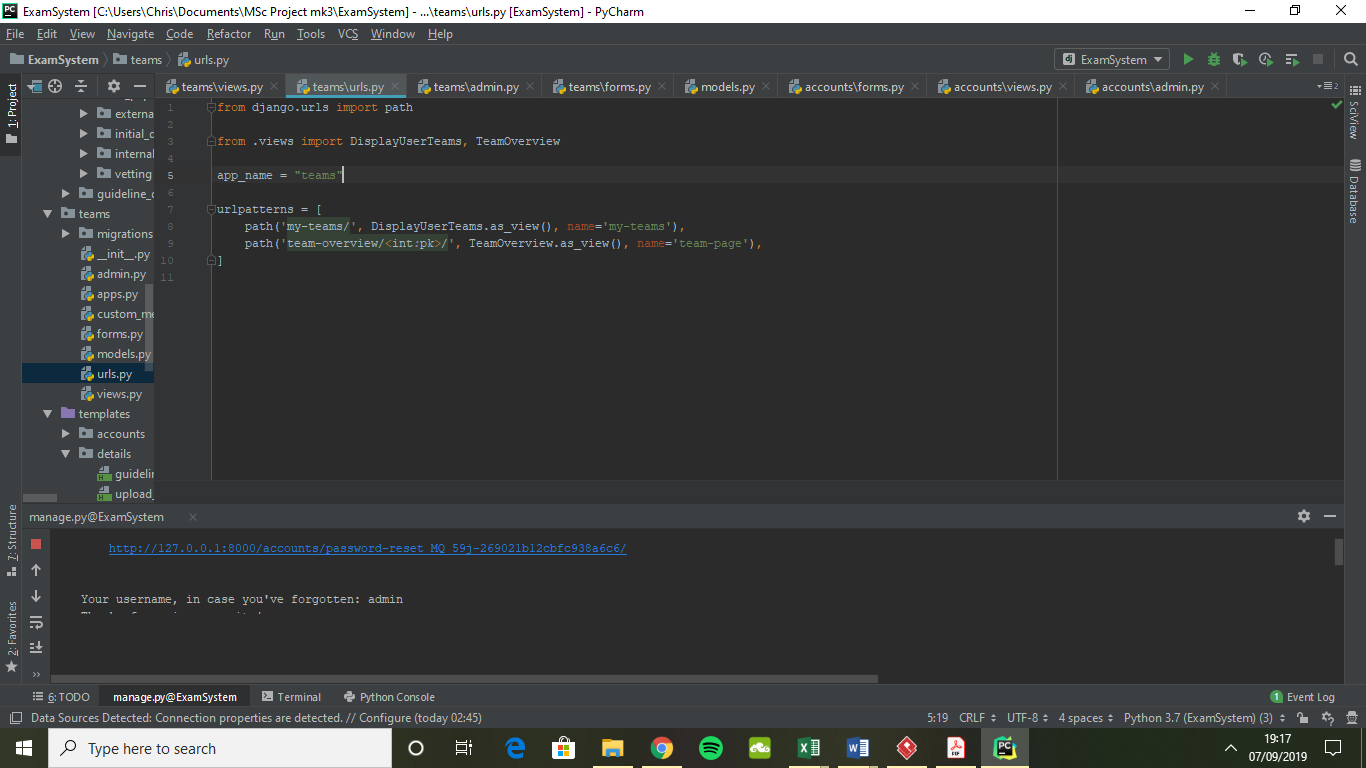
|  |  |  |
| --- | --- | --- |
| **Test** | **Expected Result** | **Expected Result Achieved?** |
| Check against the database data that all teams which the signed in user is a member of are displayed and in alphabetical order | Users teams displayed in alphabetical order | Yes |
| Check that all implemented links perform redirect the user to the appropriate page | Appropriate page loads | Yes |
| Check that no team are displayed for a user which is not assigned to a team | No teams displayed | Yes |
| Perform the above tests with users of each of the different roles, including an internal examination officer | Expected results as per specific tests | Yes |
| Attempt to access this page without being signed in | Redirect to login page | Yes |

**Team Homepage**

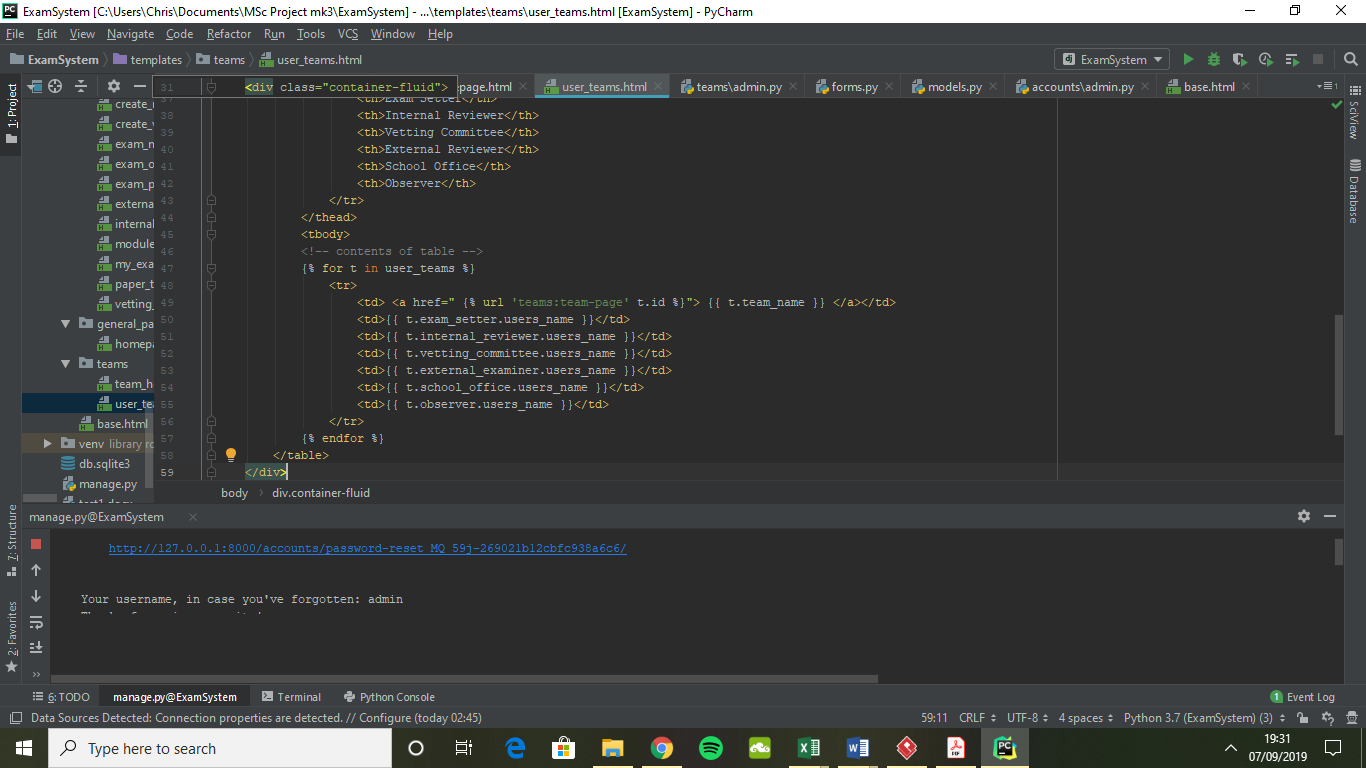
As each team can conduct the review process of multiple examinations, the decision was taken to create team specific homepages. These pages clearly show a list of all exam papers attached to the team, in addition to the relevant roles and contact detail of each member. This inclusion of team members to this screen shows an addition to the initial wireframe design (**INSERT REF)** and was included to aid the user in identifying the relevant roles distributions of the team on this page. A screenshot of the implementation is shown in **FIGURE XXXXX** at its state during the team’s sprint.



An issue that arose in the creation of this page was determining how the system would know which team page to load when it was rendering a the TeamOverview class view. After reviewing Django’s documentation (<https://docs.djangoproject.com/en/2.2/topics/http/urls/>) a solution was identified whereby the URL for the team overview page would include a unique identifier which can be used to determine which team it relates to. To implement this, a variable team identifier was added to the team URL patterns, as shown in **FIGURE XXXXXXX**.

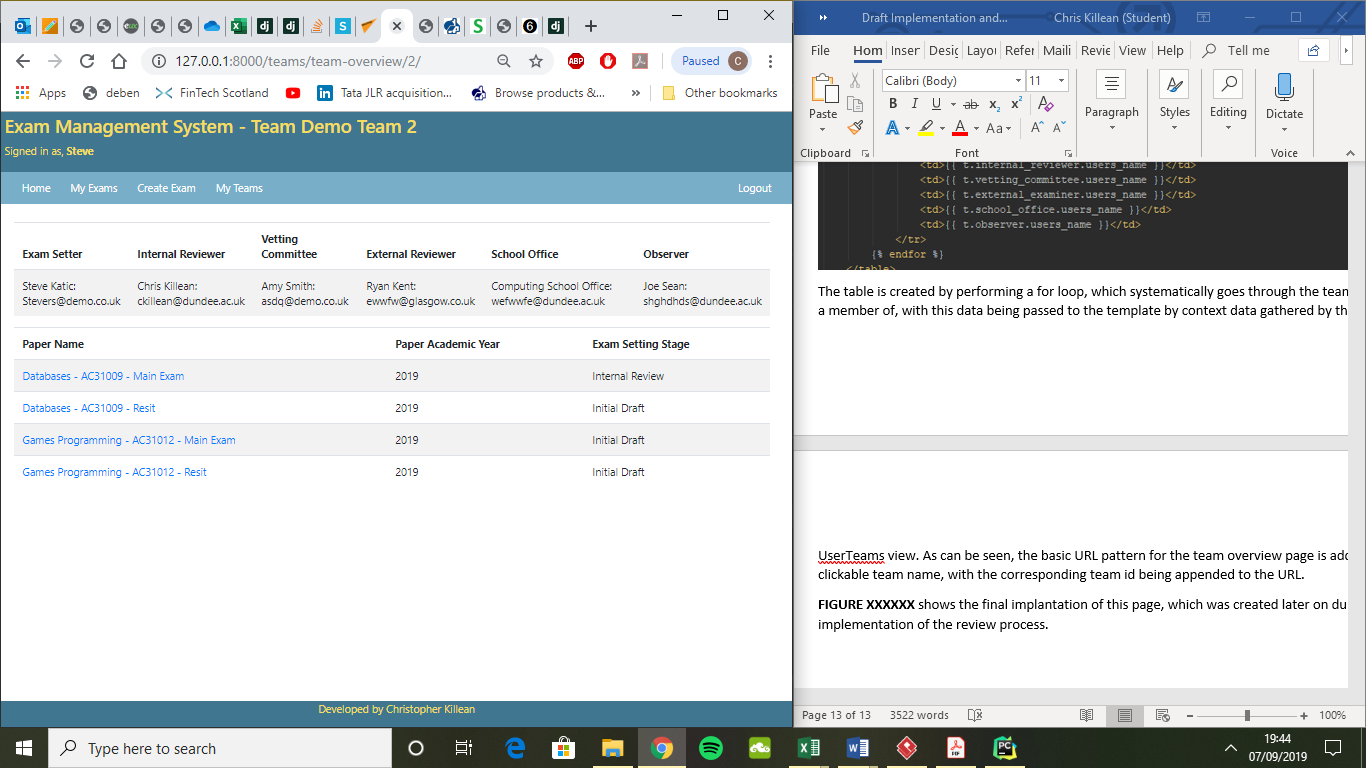


With this implementation Django would recognise any URL pattern with ‘team-overview/’ followed by a numeric value, with this value being a specific Team objects identifier. It would then attempt to render the TeamOverview view. To create unique URL links on the previous ‘My Teams’ page the developer implemented the following solution to the team list table on the page’s template.

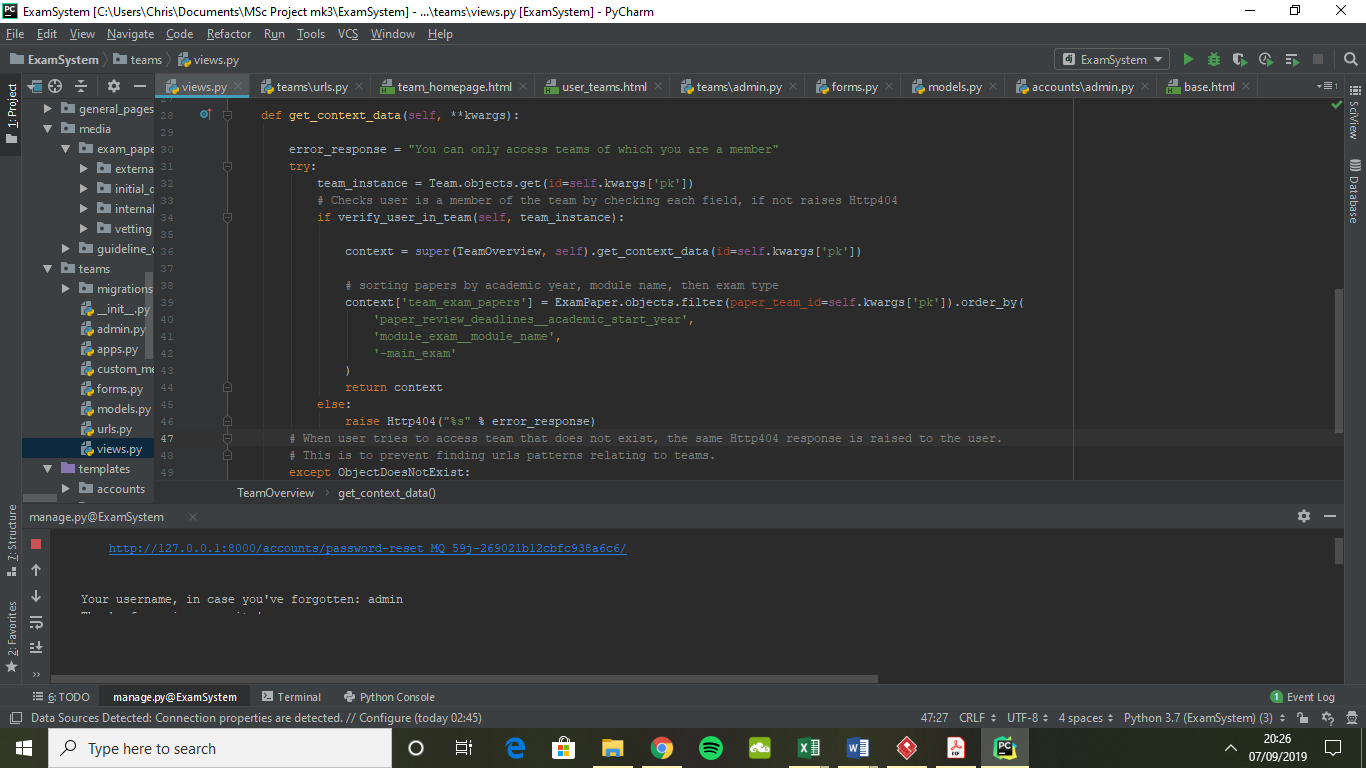


The table is created by performing a for loop, which systematically goes through the teams a user is a member of, with this data being passed to the template by context data gathered in the UserTeams view. As can be seen, the basic URL pattern for the team overview page is added to the clickable team name, with the corresponding team\_id being appended to the URL.

**FIGURE XXXXXX** shows the final implantation of this page, which was created later during the implementation of the exam review process.



The key difference is the addition of the exam papers added to the list display. This was not possible prior to the implementation of the exam review process as there were no exam papers to populate the list. Consideration was given to the ordering of how these papers should be displayed, with **FIGURE XXXXXXX** showing the adopted ordering method.



Context defines data which is passed to the template. This algorithm identifies ExamPaper objects which the team is responsible for by using the unique identifier passed in the URL pattern. These results are then ordered by the academic year of the paper (newest first), the module they are for and then by main/resit papers. This results in a user-friendly layout, so that desired papers can be easily located.

**Client Demonstration**

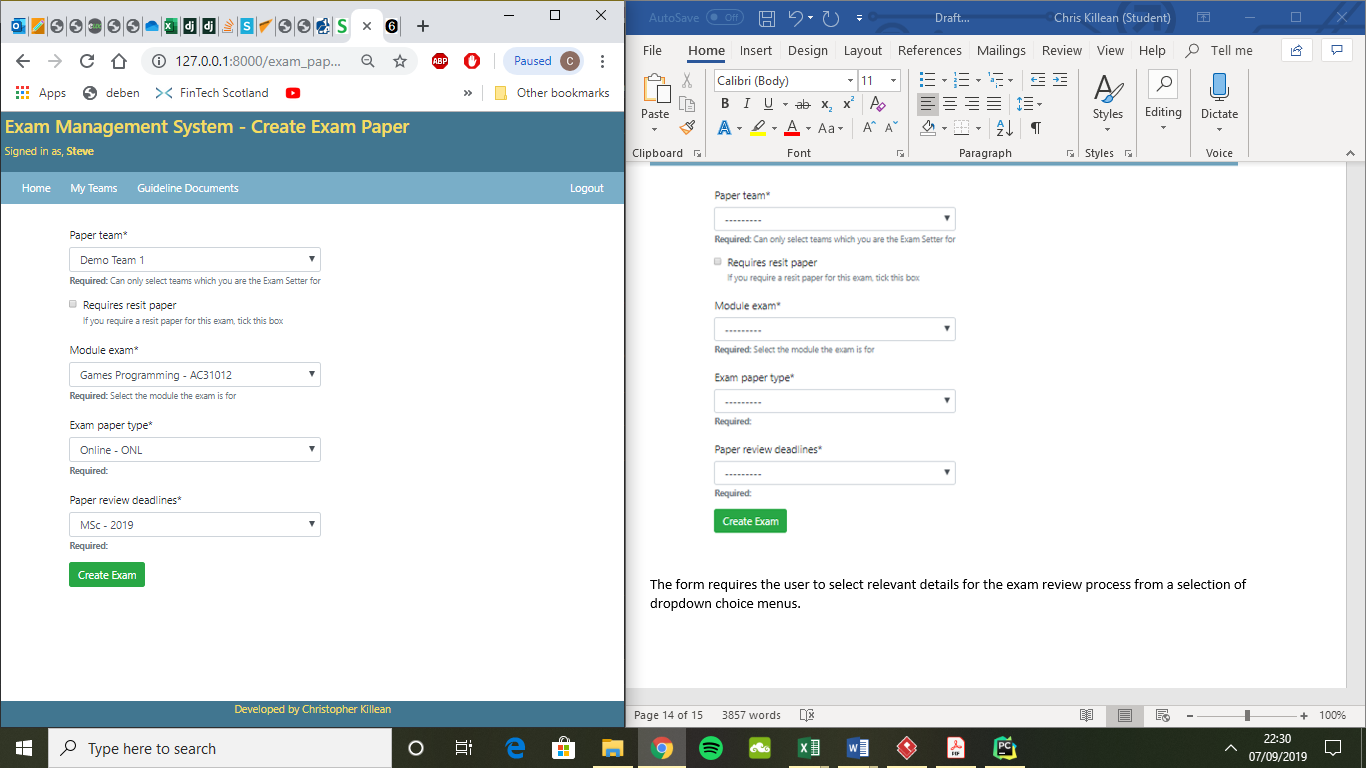
This functionality was demonstrated to the client. The client was not initially clear how the team functionality played a role within the review process. After discussion and a detailed run through the client mentioned that they could see how it may be of use, particularly if teams are given appropriate names. The client also noted that presently the review committee member is not always assigned at the creation of review teams, however this is something that could be changed to accommodate EMS.

**Exam Review**

This section details the implementation of the exam review functionality of the system. This functionality is the core purpose of the system. However, the creation of accounts and teams were rated higher in terms of importance due to essential security needs and laying the foundations of the review process.

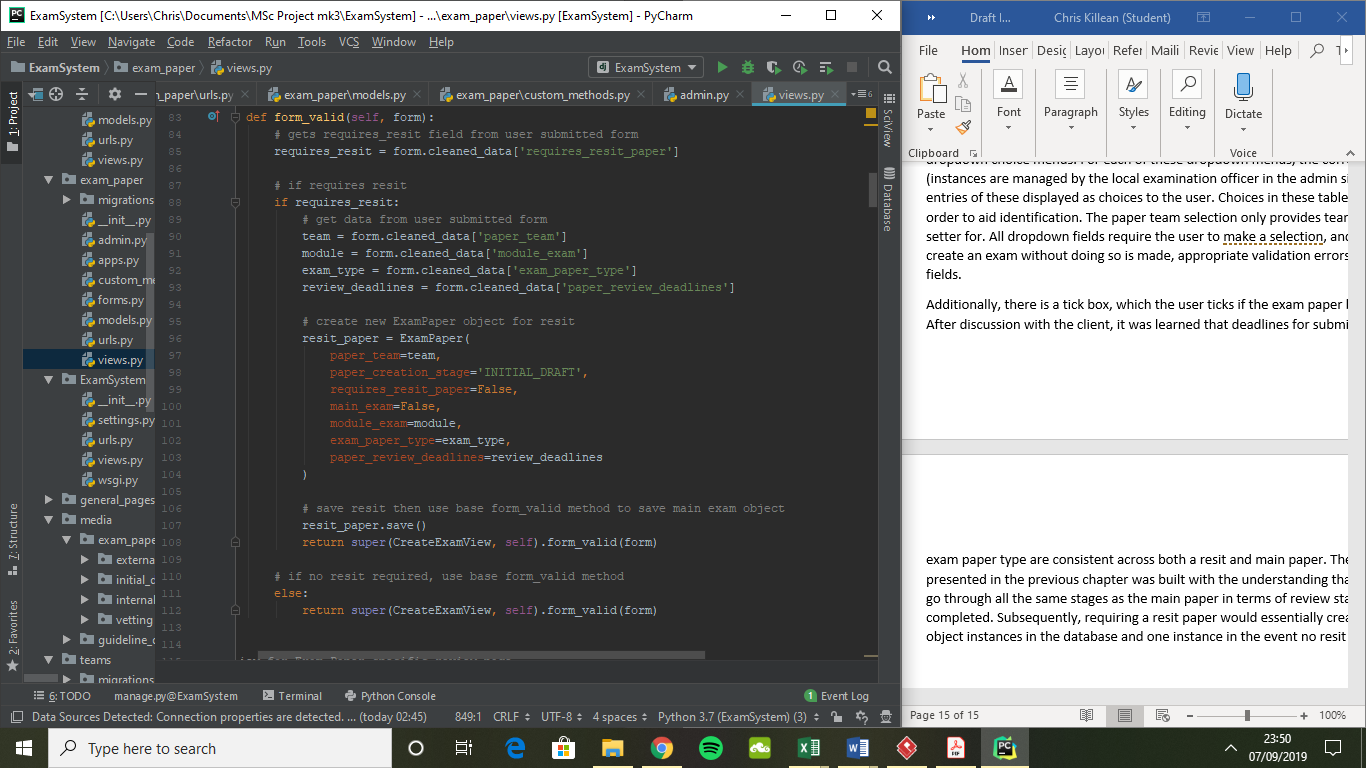
**Creating Exam Papers**

Within EMS the creating of an exam paper is the first stage of the exam setting and review process. Users that have the role of internal exam staff can access the create exam paper page of the site, which is shown in **FIGURE XXXXXX**.



The form requires the user to select relevant details for the exam review process from a selection of dropdown choice menus. For each of these dropdown menus, the corresponding database tables (which are managed by the local examination officer in the admin site) are queried, with the entries of these displayed as choices to the user. Choices in these tables are displayed in alphabetical order to aid identification. The paper team selection only provides teams which the user is an exam setter for. All dropdown fields require the user to select an option, and in the event an attempt to create an exam without doing so is made, appropriate validation errors are displayed for the unfilled fields.

Additionally, there is a tick box, which the user ticks if the exam paper being created requires a resit. After discussion with the client, it was learned that deadlines for submission of key stages and the exam paper type are consistent across both a resit and main paper. The design of the database as presented in the previous chapter was built with the understanding that a resit paper would need to go through all the same stages as the main paper in terms of review stages and reports to be completed. Subsequently, requiring a resit paper would essentially create two separate ExamPaper object instances in the database and one instance in the event no resit paper is required.



To achieve this functionality, the form\_valid function for the CreateExamView is altered. This function was chosen to be overridden as it is carried out prior to the saving of a new object to the database. The requires\_resit field is accessed and if no resit paper is required, the base form\_valid method is run. However, if a resit paper is required, the dropdown menu fields data is collected and assigned to a new object. Requires\_resit\_paper is set to false, so this method is not run again after creation, with main\_exam set to false signifying the object is a resit paper. The resit object is then saved, and then the base form\_valid method is run to perform its base method on the main exam paper object.

It should be noted that some aesthetic changes were made to this page in comparison to the wireframe design discussed previously. This is as the developer believed the resit paper option should have a greater degree of prominence on the form, and as a result this was moved further up the form.

**Reviewing Examination Material**

Each exam paper must go through a series of reviews before they can be used in student assessments. Each review stage is assigned to a specific user within a team, with only that individual able to access and complete the stage of the review process. This process has a series of review stages, with these being completed systematically. The review process adopted in EMS follows the following stages:

1. Exam setter submits an initial draft of marking material (exam paper and marking scheme).
2. Internal reviewer reviews these materials and provides a written response by means of a textbox.
3. Exam setter makes amendments based on the feedback received from the internal reviewer, documents the changes made in a textbox and uploads the most recent exam materials.
4. The materials are then reviewed by a member of the vetting committee, who provides feedback in a textbox.
5. Exam setter makes changes based on this feedback and uploads most recent exam material.
6. External examiner reviews these materials and provides feedback based on them in an onscreen textbox.
7. Exam setter makes changes based on this feedback and uploads the amended exam materials. The exam material at this stage is ready for students to undertake the exam.
8. After students have sat the exam, the exam setter completes the exam markers report.
9. Finally, the internal reviewer completes the module moderators report. The review process is complete at this stage.

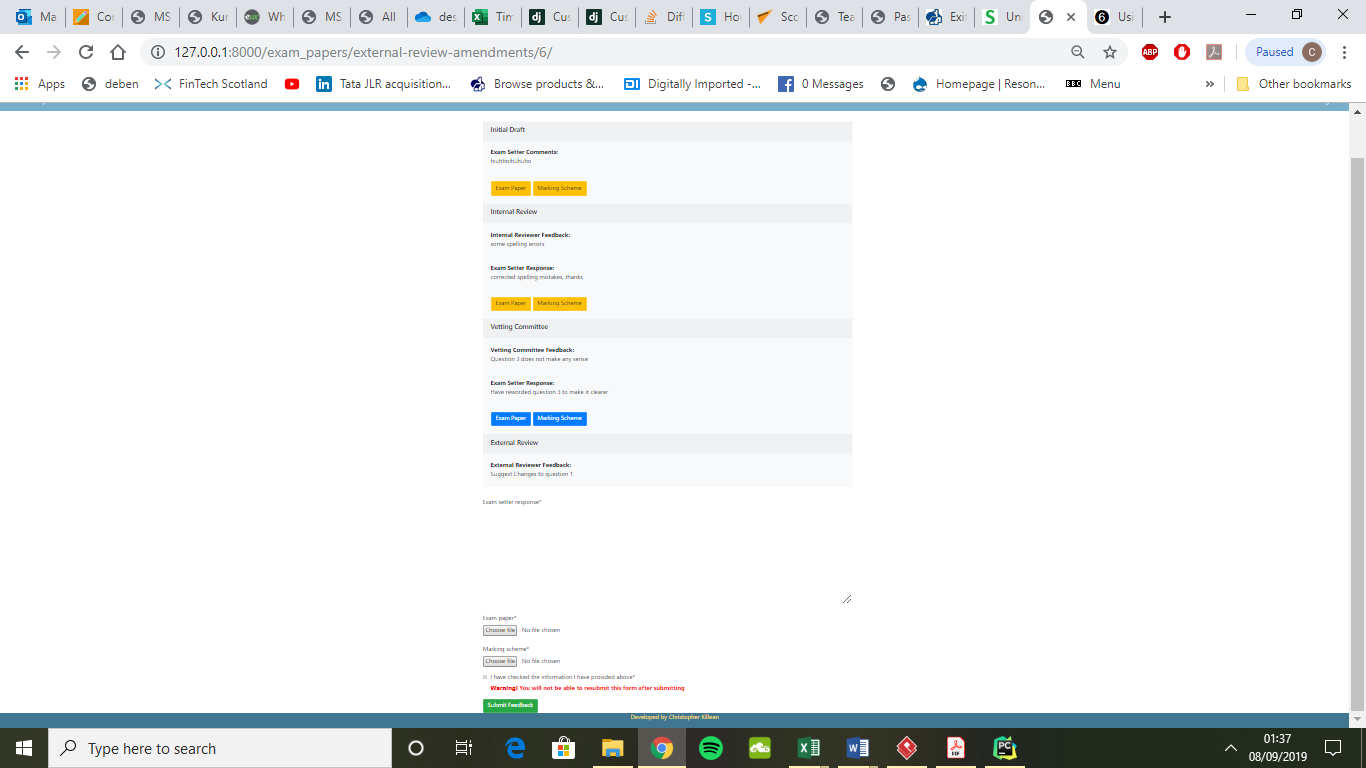
**FIGURE XXXXX** shows the form the exam setter completes on EMS after the external examiner has reviewed the examination material. Each of the review stages prior to this stage have a similar layout, with the main differences being:

* Prior stages will have fewer comments and stages displayed, as they will not have been completed yet.
* When an individual is reviewing the exam setters submitted documentation, their screen does not display fields for file uploads. Instead they submit feedback via the textbox.

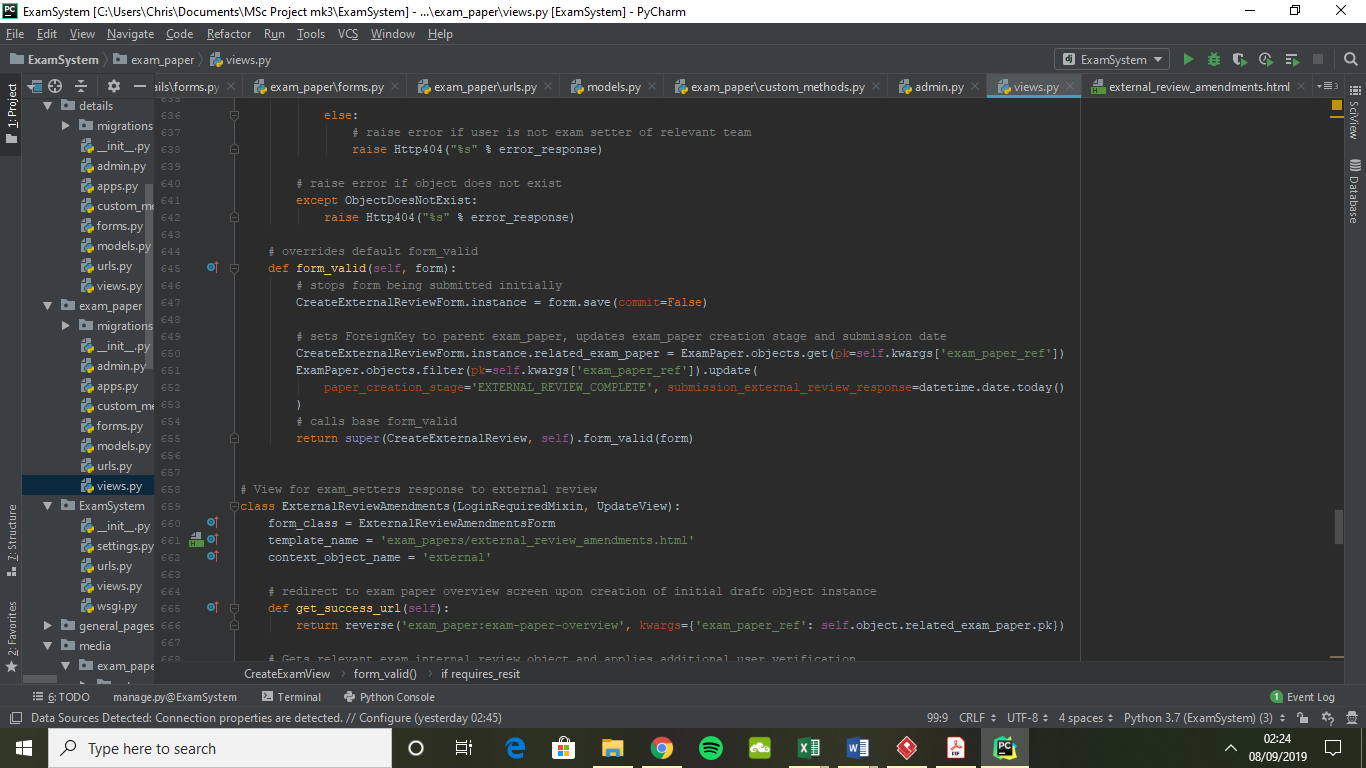
As can be seen, previous feedback, exam setter responses and uploaded files are available to the reviewer. Initially, the developer had consider having no previous feedback due to the possibility of group-think negatively impacting the quality of reviews if prior feedback had been very positive or negative (Shirey, 2012), with this concern being raised with the client. The client suggested that presenting a complete audit trail up to the point of review could improve the quality of feedback as the reviewer could check to see if past feedback had been acted upon, or why changes had been made. Subsequently, the developer implemented this functionality into EMS.

When presenting the file downloads of previous review stages, the developer sought to include a visual que of which ones are from previous stages and which are the most recent. This was done by making the link to the most recent blue, with previous stages displayed in yellow.

The way the system is implemented, once a review stage has been completed, it cannot be returned to for editing. To make this clear to user, a tick-box was added which must be pressed prior to submitting the form. This tick-box is accompanied by help text displayed in red to draw the user’s attention to it. Additionally, to reduce potential user errors of submitting incomplete forms, all fields have relevant validation, for example file upload fields must have a file uploaded and textboxes must have text entered otherwise error messages are raised when the form is submitted.



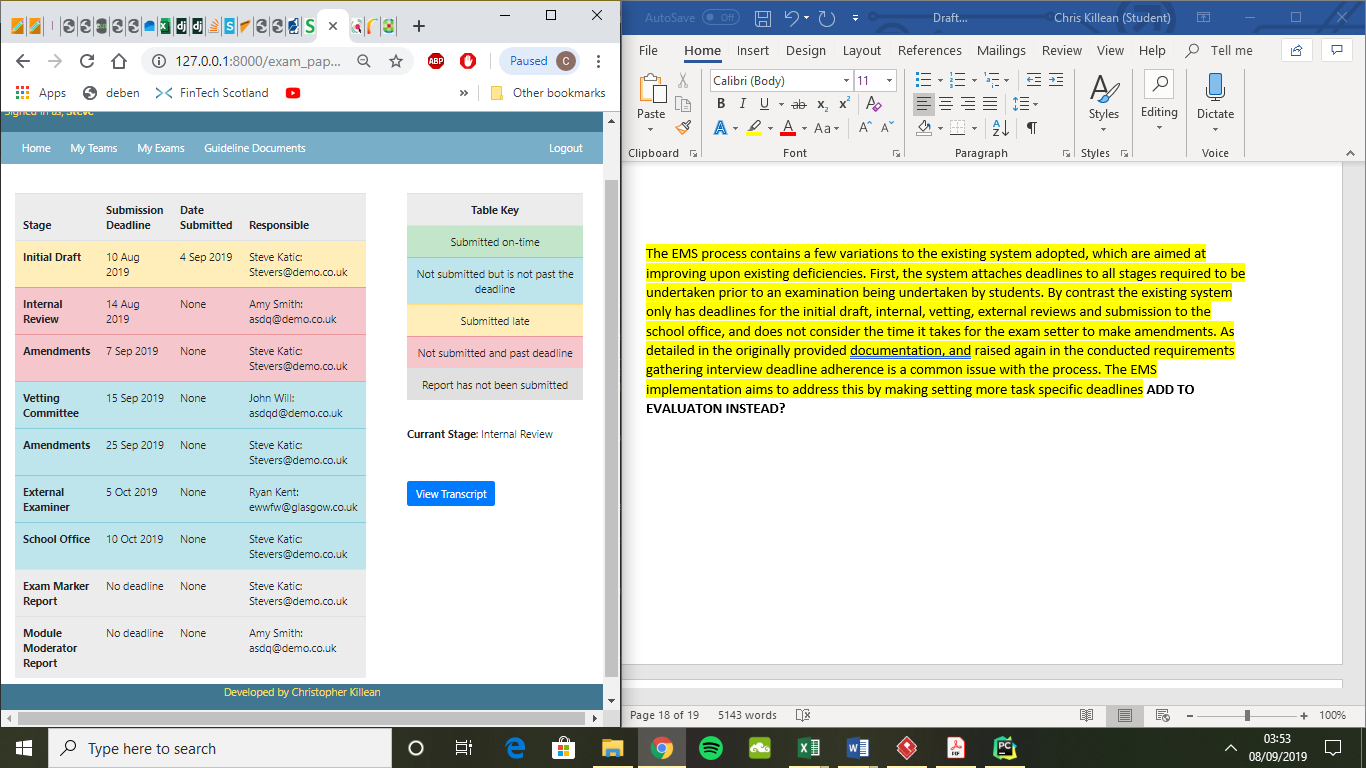
When implementing the review stages, the developer had to ensure that when a form was successfully submitted, fields relating to the date of submission and the review processes present stage were updated in the relevant ExamPaper object. This posed as issue, as the view class CreateExternalReview would not by default implement these fields in a foreign table.



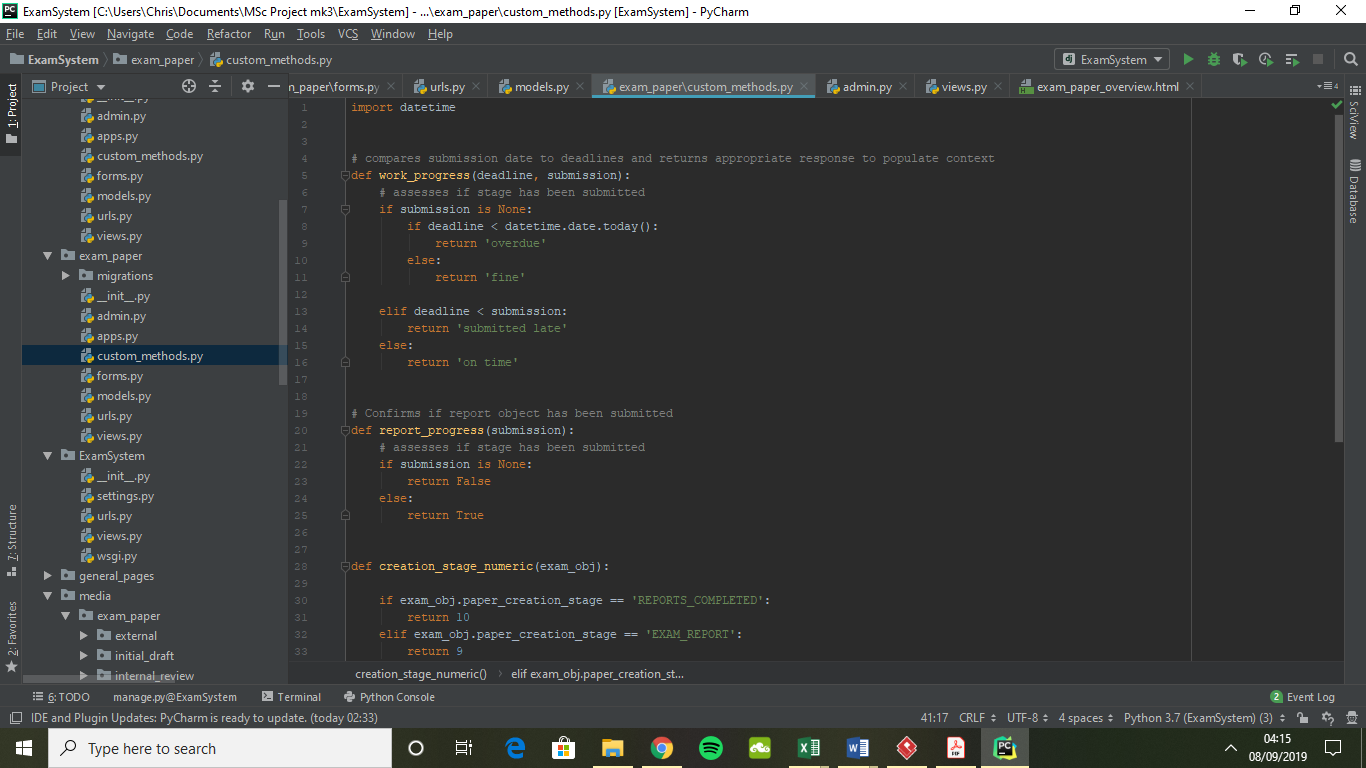
To overcome this, the developer chose to alter the classes implementation of the form\_valid method. First the process for calling the save method is stopped to ensure the form does not initially save. Then the form that will create the ExternalReview object has its related\_exam\_paper foreign key field set to the appropriate ExamPaper object, with this being identified by a unique key contained in the URL. Next the fields paper\_creation\_stage and submission\_external\_review\_response in the ExamPaper object are updated to reflect the date the external review was submitted and the present stage the exam paper is in the review process. Finally, with these fields set, the base form\_valid method is called to complete the creation of the ExternalReview object. Similar implementations are applied for the other review stages.

**Exam Paper Overview**

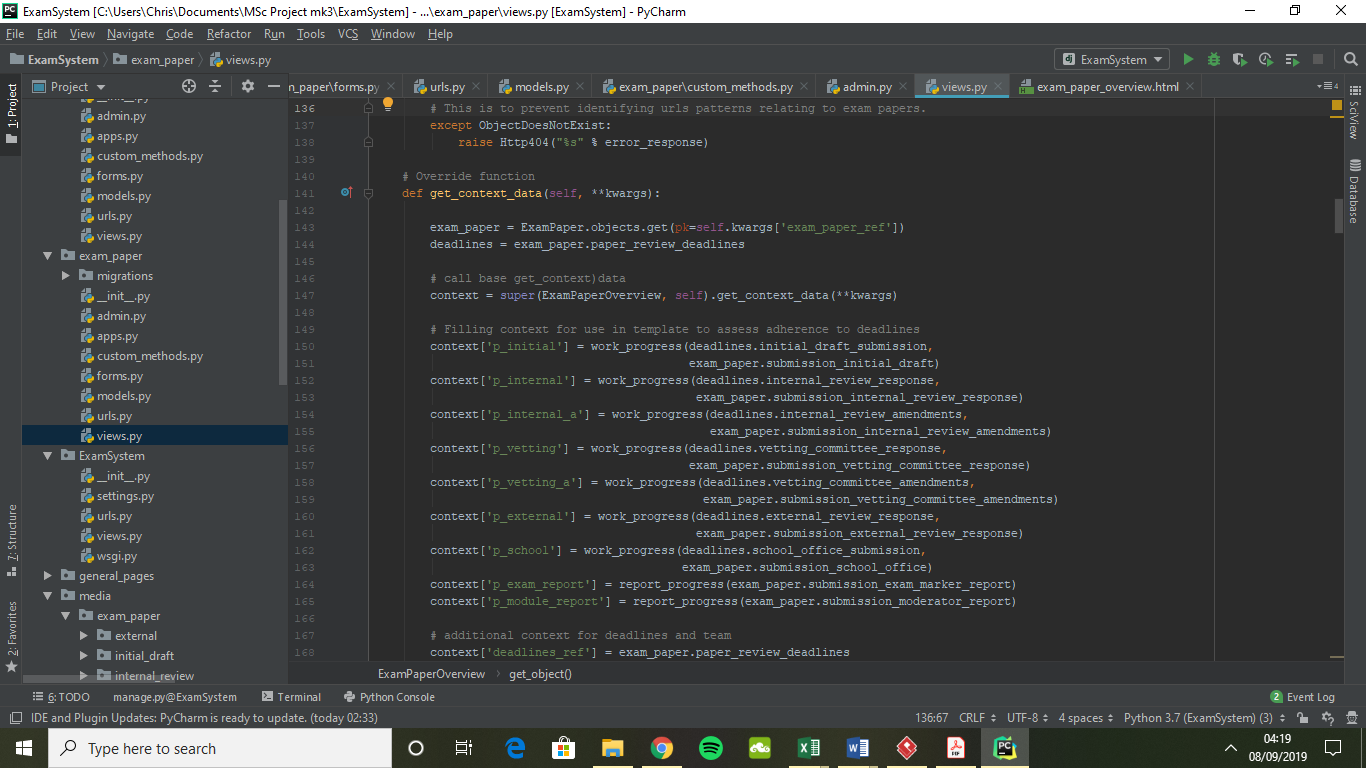
Within EMS, each exam paper has its own unique homepage. This homepage gives an overview of the exam papers progress through the review process. This is implemented in an attractive design which clearly indicates the present review stage, who is responsible for each review task and each stages adherence to submission deadlines. Additionally, there is a ‘View Transcript’ button which when clicked takes the user to a screen that shows past review stages and if they have been submitted, the Exam Marker and Module Moderator reports. If the presently signed in user is responsible for the current stage review task, a button is presented which takes them to the page where they can complete this task. **FIGURE XXXXXXX** shows an example of this page.



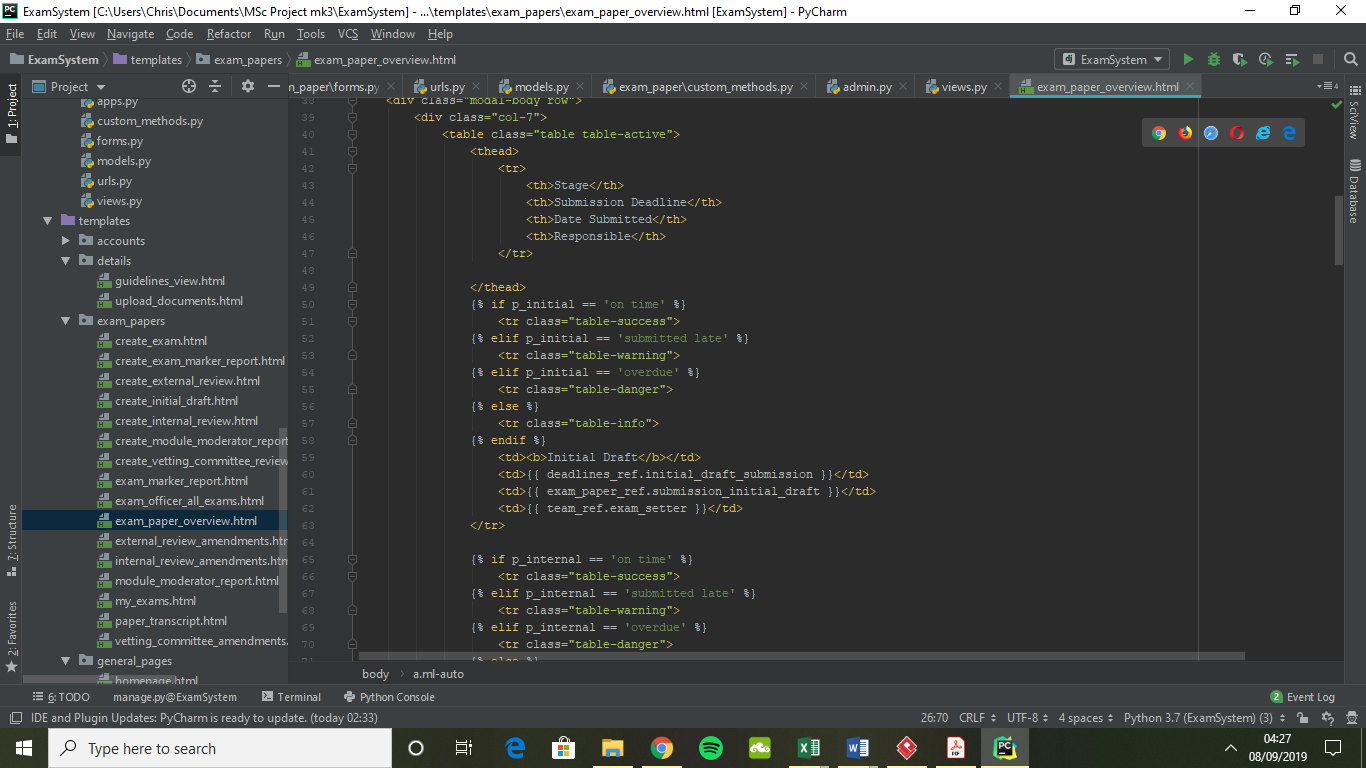
When the main table was first implemented the view template from a development point of view became difficult to read as much of the logic was implemented in the template. This was primarily due to submission deadlines and the date a review stage was submitted being spread over multiple tables in the database schema. To combat this the developer adjusted the solution, so that a greater degree of the logic was carried out in the ExamPaperOverview class view.



First the developer created a custom method which would compare the deadline of a review stage and when it had been completed. A string value is then be returned indicating if the stage was overdue, submitted late, submitted on time, or had not been submitted but is not past its deadline.



Next, the ExamPaperOverviews views get\_context\_data method was changed (**FIGURE XXXXX)**. First the relevant exam\_paper and deadlines objects were collected. The base get\_context\_data was then called, which would gather the default context data for this particular view. Additional context data was then appended, which would each be populated with the string output from the custom method work\_progress.



Subsequently the template became much more readable (**FIGURE XXXXXX**), while also carrying out significantly less logic. The above template extract demonstrates both the population of data for the initial draft row in the team overview table, with minimal logic used to determine the colour of the row.

The EMS process contains a few variations to the existing system adopted, which are aimed at improving upon existing deficiencies. First, the system attaches deadlines to all stages required to be undertaken prior to an examination being undertaken by students. By contrast the existing system only has deadlines for the initial draft, internal, vetting, external reviews and submission to the school office, and does not consider the time it takes for the exam setter to make amendments. As detailed in the originally provided documentation, and raised again in the conducted requirements gathering interview deadline adherence is a common issue with the process. The EMS implementation aims to address this by making setting more task specific deadlines **ADD TO EVALUATON INSTEAD?**

Shirey, M. R. (2012) ‘Group think, organizational strategy, and change’, *The Journal of nursing administration*, 42(2), pp. 67–71. doi: 10.1097/NNA.0b013e3182433510.