The purpose of the project was to produce a system which will allow the students, secretary and the scrutiny panel to efficiently and securely store and access Extenuating Circumstance Forms as well as allow other staff members to access the sanitised version of the students circumstance. Django provided a simple and secure platform with the ability to access the system on any device unlike the alternates of Smart Applications and Blockchain. Smart applications would not be accessible on all devices and Blockchain would not be able to keep the data extremely confidential. Each of the following describes the requirement as well as the results achieved and why the use of Django was the right choice.

Organised Data

One of the issues with the previous system was the processing of data. A lot of data would be need to be duplicated which lead to unorganised data records. Students would submit multiple forms at times with either a different circumstance or the same one which would again lead to improper data storage. With the Django system the data is well organised and easily accessible. Data is stored in tables with the ability to search for specific records and view past submissions as well. Figure 1 and 2 show extenuating circumstance forms for both Students as well as what the Secretary and Scrutiny Panel would see. The status of the form allows all stakeholders to remain informed and students avoid multiple forms filled with the same data. The search field allows sorting and filtering through the forms extremely easily when searching for students, specific forms or status search. This solves the problem of data being unorgainsed.

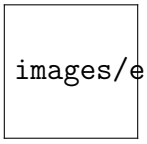

Figure 1: Student Panel Submitted Forms

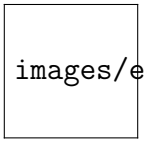

Figure 2: Secretary/Scrutiny Panel Submitted Forms

Additional Files

In situations where there secretary and scrutiny panel decide they need to get more proof and data from the student, the secretary can request files via the panel; automatically sending an E-mail notification to the student requesting them to submit more proof. The student can then access the form on the panel and upload the files, on successful upload another E-mail is sent to to the Secretary confirming that the student has uploaded more proof and should be taken to the next

Scrutiny Panel meeting. This provides the student and the secretary simplicity as any new proof is directly linked to the form. With a web based platform like Django, using IF statements made such tasks extremely easy to code with. Figures **??** and **??** show the section of additional files under the student panel.

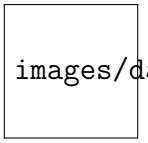

Figure 3: Dashboard notification for more files

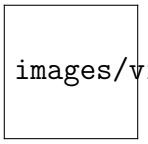

Figure 4: View Form upload files

Print, Download & Student Record

Being able to share the data to other third parties was an important requirement as well as being able to get it on paper. With the use of JavaScript, printing and saving as a PDF document was implemented successfully with the output being exactly the same as what a user would view on the website as explained in Chapter **??**. Figure **??** shows two simple buttons which have the JavaScript code behind it to simply print or download the current page in the same format it is viewed.

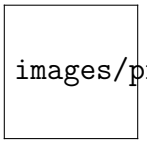

Figure 5: Print and Download Files

Another requirement was being able to share the students record where the sanitised version of data is stored to the student portfolio. The current implementation uses a unique link to each students profile which can be added to the students portfolio in order to share the data. Only staff members will have access to the student portfolio hence access to the records. Figure **??** shows the secretary panel with the ability to copy the link or re-generate it to share with others. This regeneration process became extremely easy as Django uses *Python* which has a *random* package as mentioned under Chapter **??** allowing the link to follow a random 8 digit format.

Figure 6: Print and Download Files

Security

Security was one of the most important requirements for this system and with Django being the back-end controller it made the system secure. All-Auth[**?**] provides validation for the sign up and uses *Bcrypt* [**?**] encryption to securely store the password into the database. If the password is not complex enough, All-Auth would automatically bring up an error and ask the user to create a stronger password. The database itself is password protected when storing data, for example, *Heroku* uses a *25 digit complex password* to secure the database itself which is prevents brute-force attacks onto the database directly. The current implementation does not encrypt every single field in the database but with the use of *django-pgcrypto*[**?**] each field can be encrypted making the data even more secure than it already is.

Apart from the database security, it was important to make sure other users who can login to the system such as Students cannot access data which is specifically meant only for the Secretary or the Scrutiny Panel. This was successfully implemented with the use of FOR loops and IF statements as described in Chapter **??**.

The student record page(Sanitised version of the students circumstances) which is accessible to staff members would need to be behind *University of Sheffield's Intranet* in order to authenticate if the user is a staff member or not. Even without staff authentication, the student records page can only be accessed by having the student's key as explained under Chapter **??**. A Key system would prevent unauthorised access to the students records as the link would only be accessible in the students portfolio. In cases where the link has been shared numerous times it can be regenerated by the secretary to prevent access on the old link.

Other security tests were also completed, some provided by Django's inbuilt tests while others required manual testing. The security requirements were satisfied based on the tests ran such as *Raw SQL Queries, Cross Site request Forgery and Unittests*. However, not all code has been tested due to lack of expertise on site penetration

Lastly, uploaded files are stored directly onto the hosting server. In order to secure the files, the use of *SSH-Keys* to access the hosting server would provide increased security. Another approach, currently not implemented in the system, is to individually encrypt each file as it is stored onto the system. Two stable, but not maintained, Django encryption projects can be used. One creates a transparent layer of encryption above the files[**?**] while the other encrypts the files

based on the form input in Django[**?**]. Both provide a good level of encryption which would make all the files extremely secure and confidential.

Mobile First

Since the website is coded in Bootstrap 4[**?**] the system follows a mobile first coding approach which allows the website to be easily accessed on any mobile devices(Smart phones, tablets or laptops) with full functionality. Students can apply for extenuating circumstances at the comfort of their home in cases where their circumstance does not allow them to move out just to submit a printed form to the office. Figure 3 shows the system running on a *Heroku* Server and accessed on a mobile device as a Student trying to create a new Extenuating Circumstance Application. All panels including the Secretary Panel are mobile friendly and so can be accessed with the smallest smart phone.

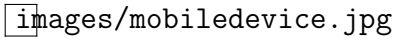

Figure 7: Screenshot of the system running on a mobile device

Email Notifications (Additional Feature)

Communication between the system and the users is an important feature to have. The implementation uses *Django's mailer system* to send E-mails to the Students as well as the Secretary on successful actions such as *new form submitted*.