School Learning Management System – Coding and Implementation

Introduction

This document covers the coding and testing of a secured learning management system for schools (Se-LMS). The Se-LMS allows teachers, students and parents to view students' assignments and grades securely. Se-LMS have been promoted in the UK over the last decade (Ofsted, 2013) while the Department for Education (DfE) require schools to provide high-quality remote education (DfE, 2024). The application, 'Smart MAT' is targeted at schools or multi-academy trusts in the UK.

This report should be read in conjunction with the README file, detailing how to run the Smart MAT application and the code has been provided in the supporting files. A number of illustrations have been used throughout and are summarised in the table below:

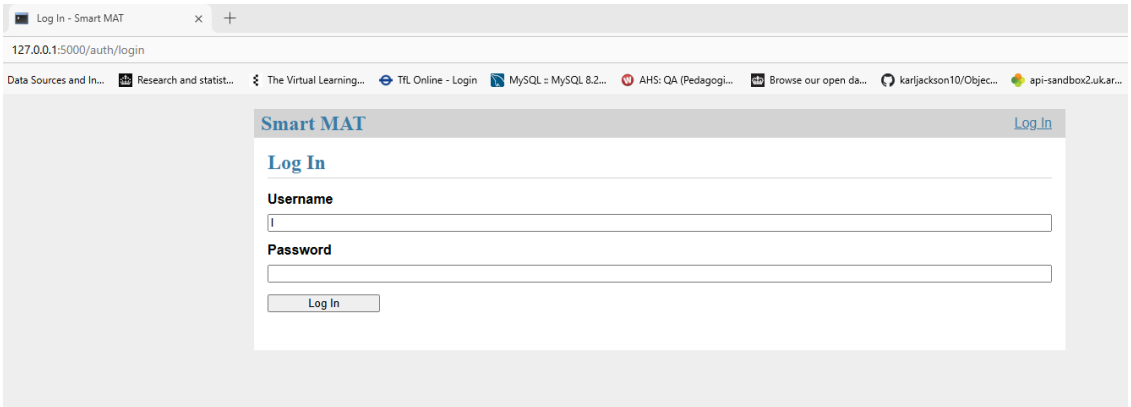
Table of Figures

The Application

The application has been designed to run with or without security. The following sections demonstrate the working application running with security turned on.

Login

Figure 1 - Login

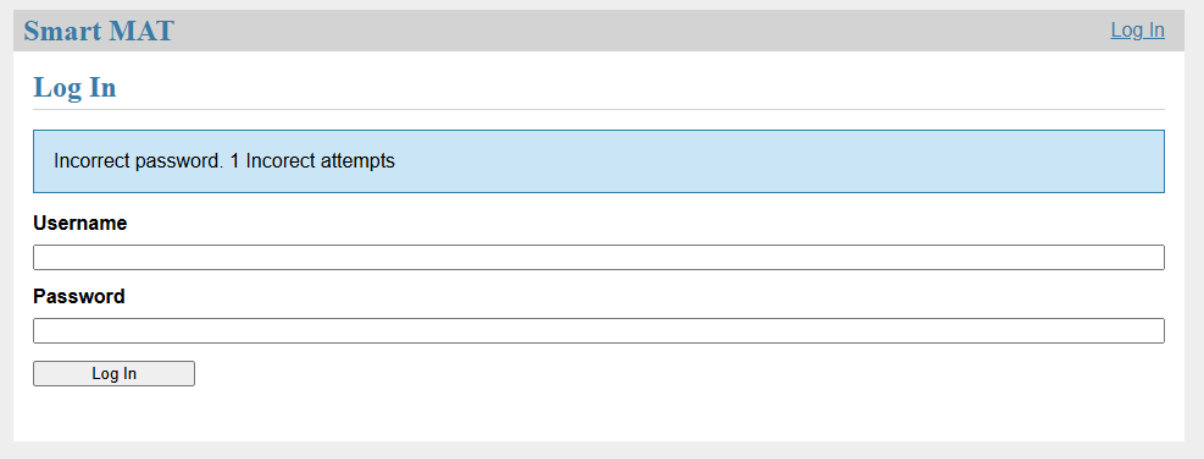


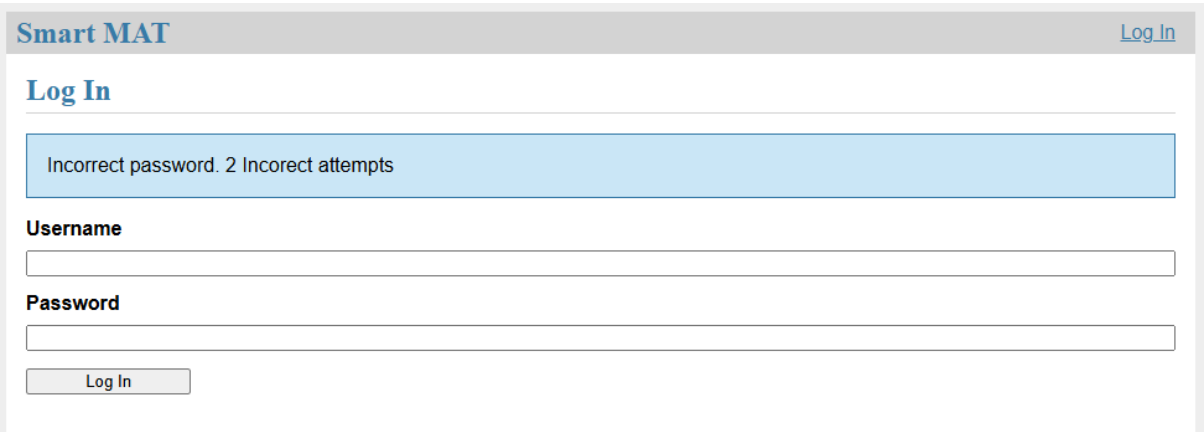
The login in page prompts the user for a valid username and password. If the user provides an incorrect password, an error is displayed, and the incorrect password counter is increased by 1. The user account is locked after 4 incorrect attempts. This prevents a brute force attack where a hacker would be able to continually

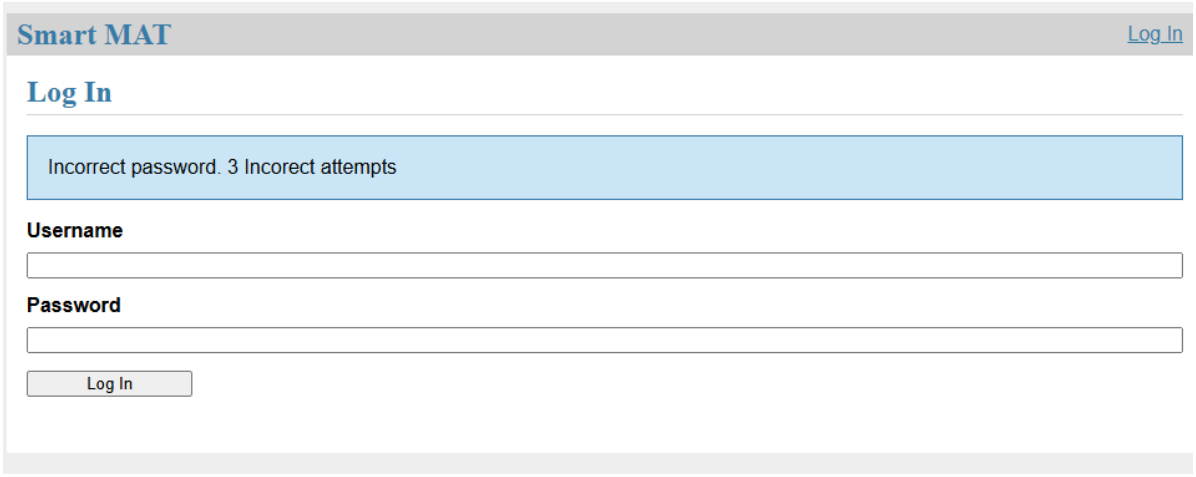
attempt to login to the application. This security feature is disabled when the security settings are turned off.

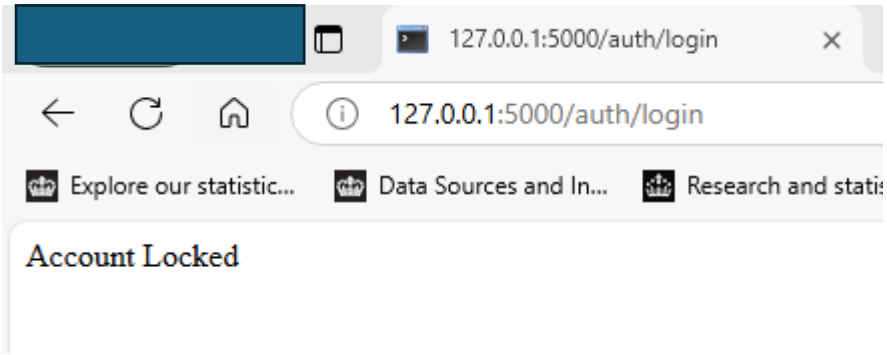
It should be noted that passwords are secure as they follow a Regex pattern and they have been hashed for additional security.

Figure 2 - Incorrect Login









Main Menu

The main menu will reflect the permissions of the user according to their role: Admin, Teacher or Student. This ensured that when security is enabled, the user is only able to access functionality relevant to their role.

Figure 3 – Admin Menu

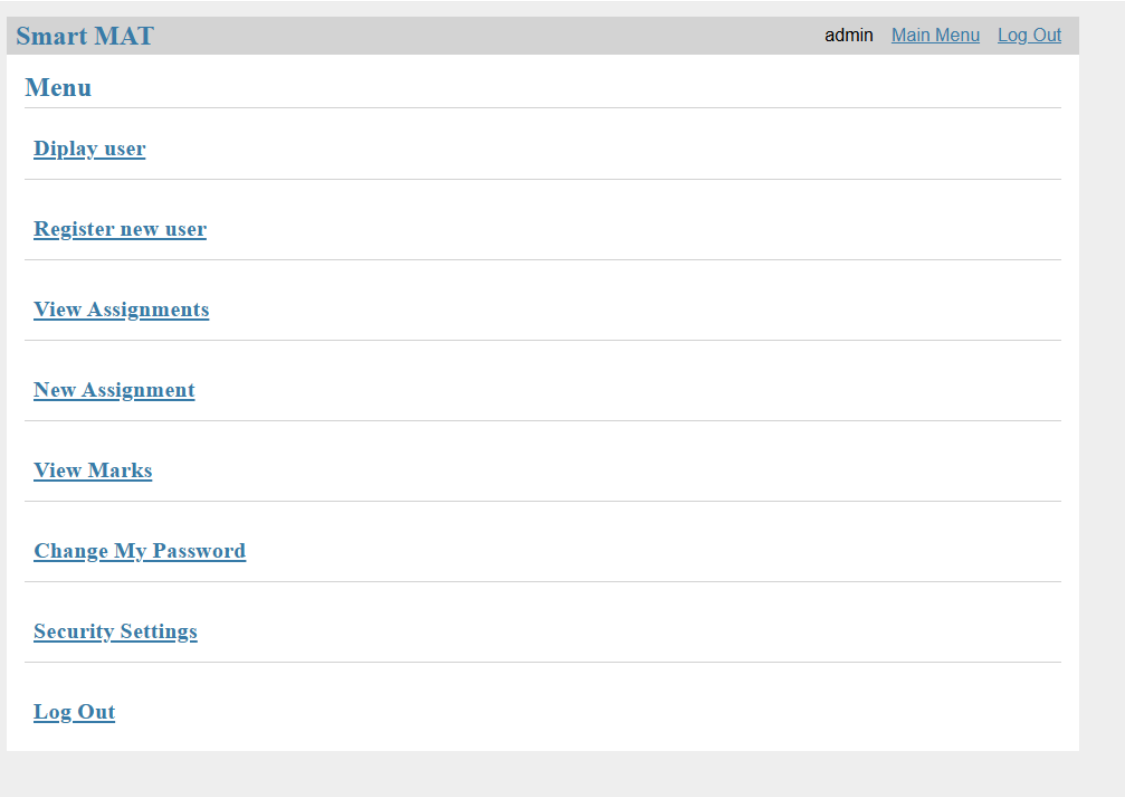


Figure 4 - Teacher Menu

Figure 5 - Student Menu

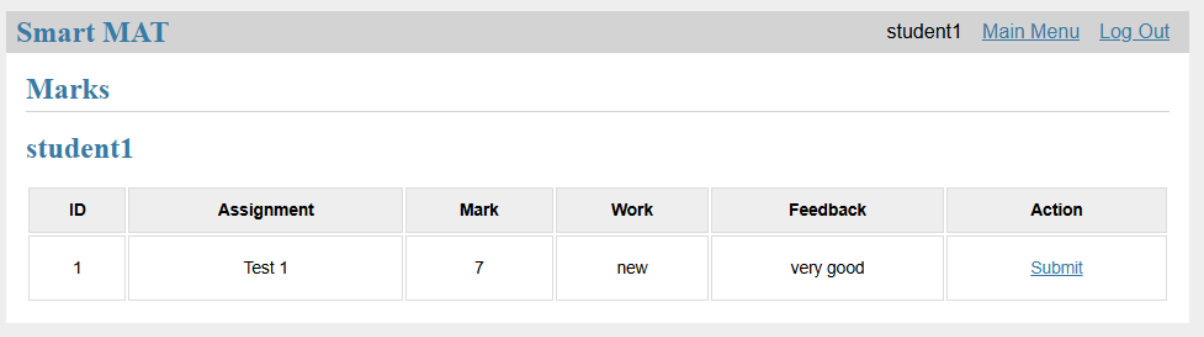
Functions by role: Students

Student access is restricted to the functions necessary for their role. Students can only see assignments that have been allocated to them

Figure 6 - View Student Assignment

Students can view their marks for an assignment

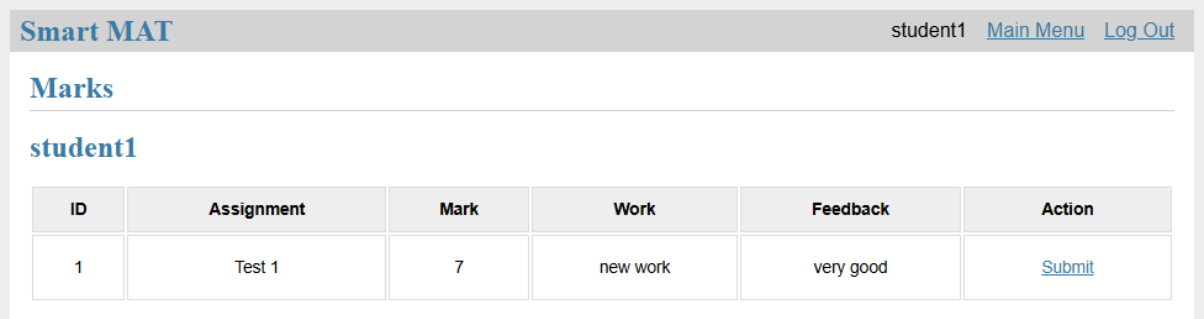
Figure 7 - Student View Marks & Submit Work



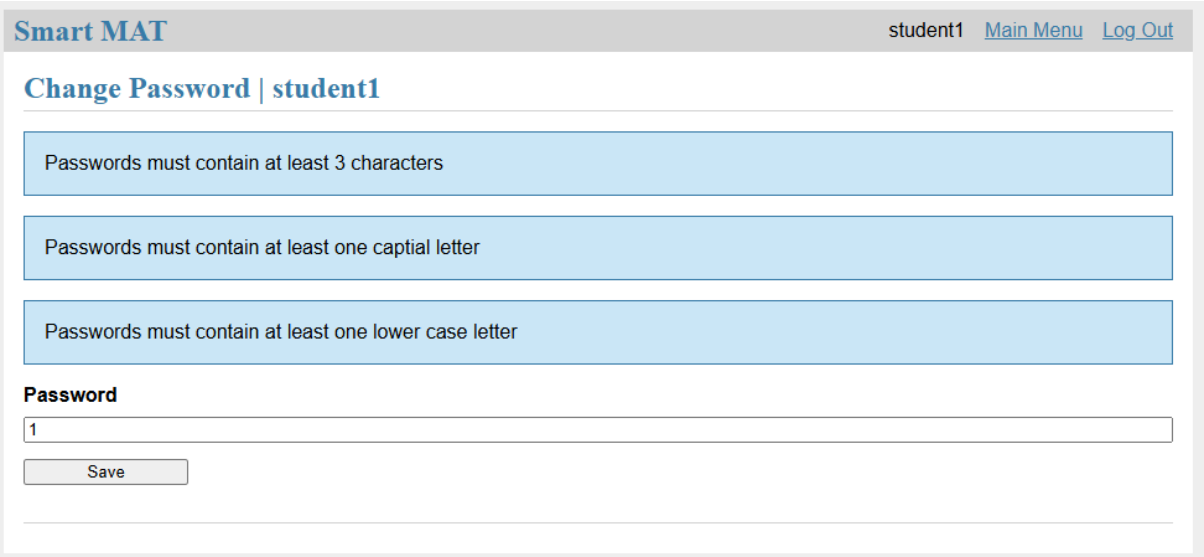
Students are also able to submit work via a form. There is no opportunity to upload files which helps to prevent sql injections.

Figure 8 - Submit Assignment

Figure 9 - Updated Submission



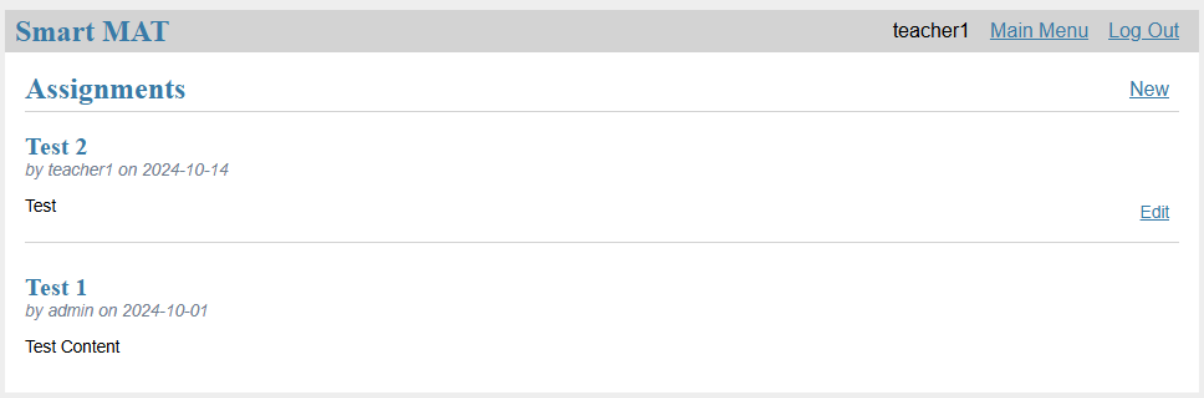
Users of any role can change their own password, if the new password meets the security requirements of the Regex pattern.



Functions by role: Teachers

Teachers can view all assignments but can only edit the assignments that they own.

Figure 10 - View Assignments - Teacher



Teachers can edit or delete their own assignments.

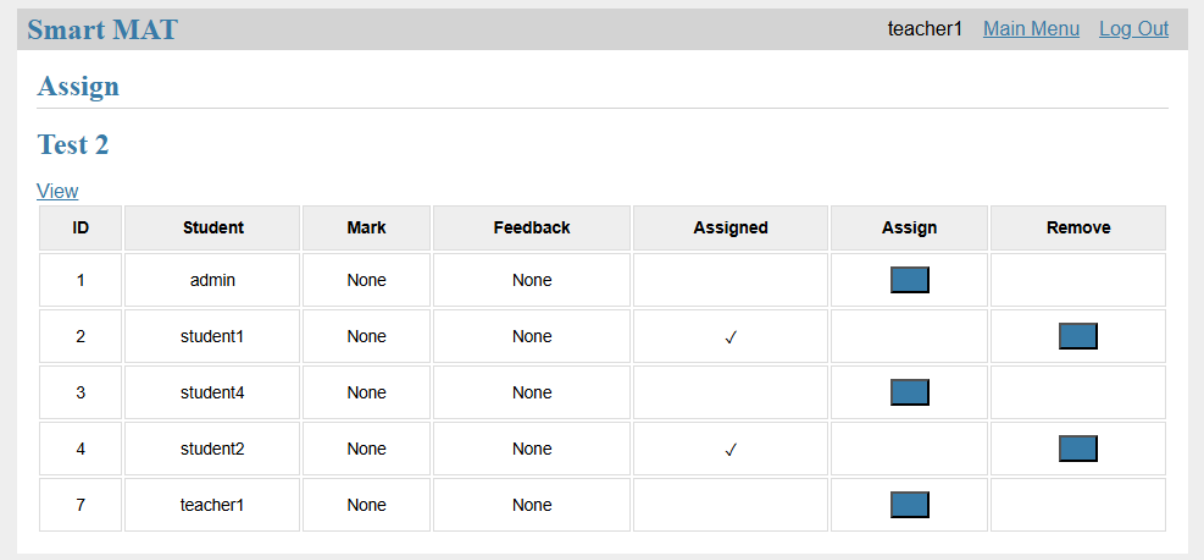
Figure 11 - Edit Assignment

Teachers can view all assignments and manage the ones they own, using either the ‘Assign’ or ‘View’ options.

Figure 12 - Assignment Marks - Teacher

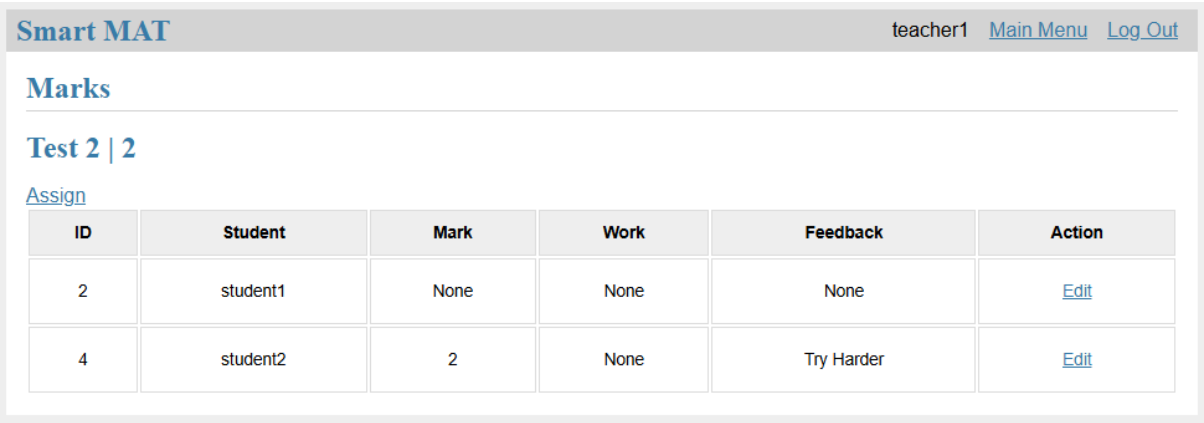
For any assignment they own, Teachers can assign or remove users.

Figure 13 - Assign users



They can also view the marks:

Figure 14 - Teacher Marks



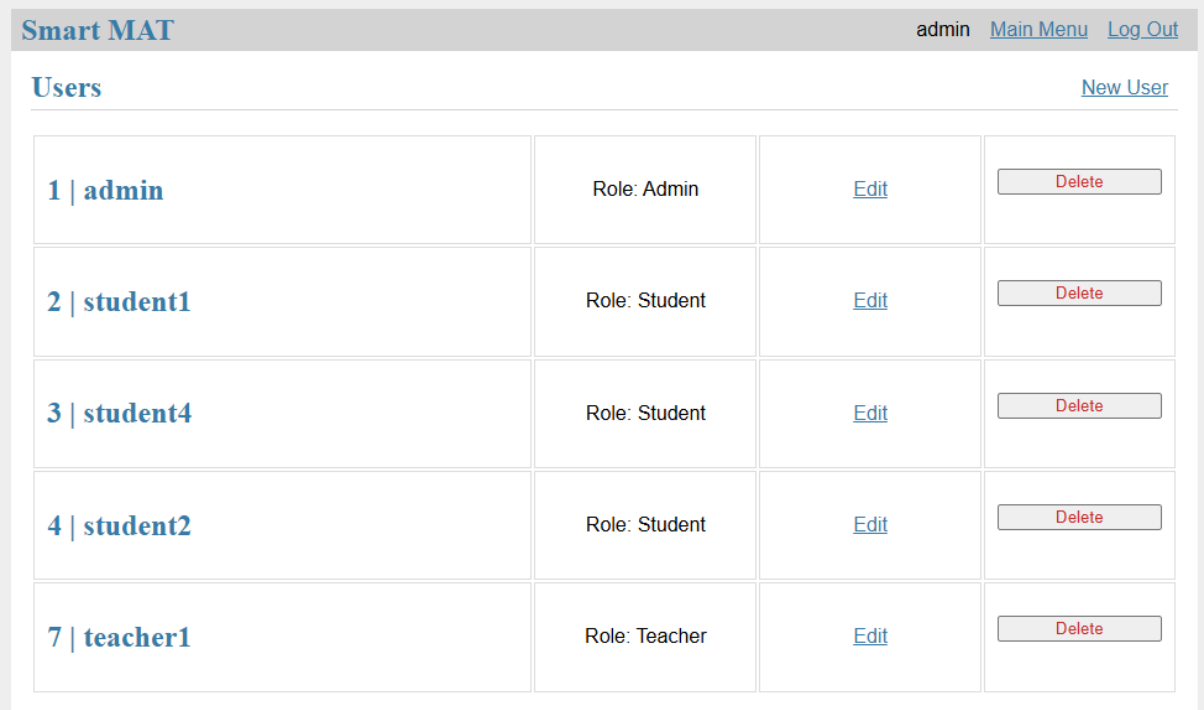
Teachers can edit the marks and feedback, for their assignments

Figure 15 - Edit Marks

Functions by role: Admin

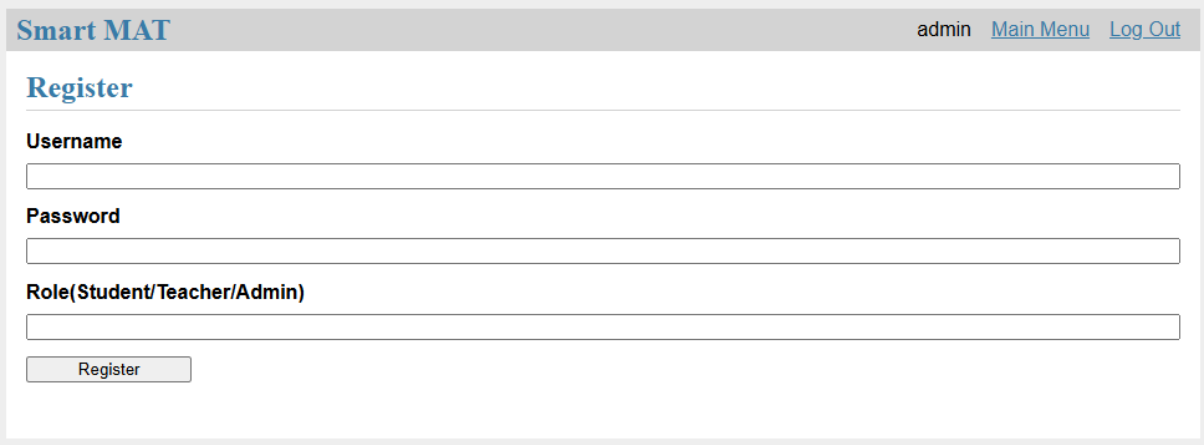
Some functions are restricted to the Admin role. Admin users are able to view and manage all users' details.

Figure 16 - Display and Edit Users



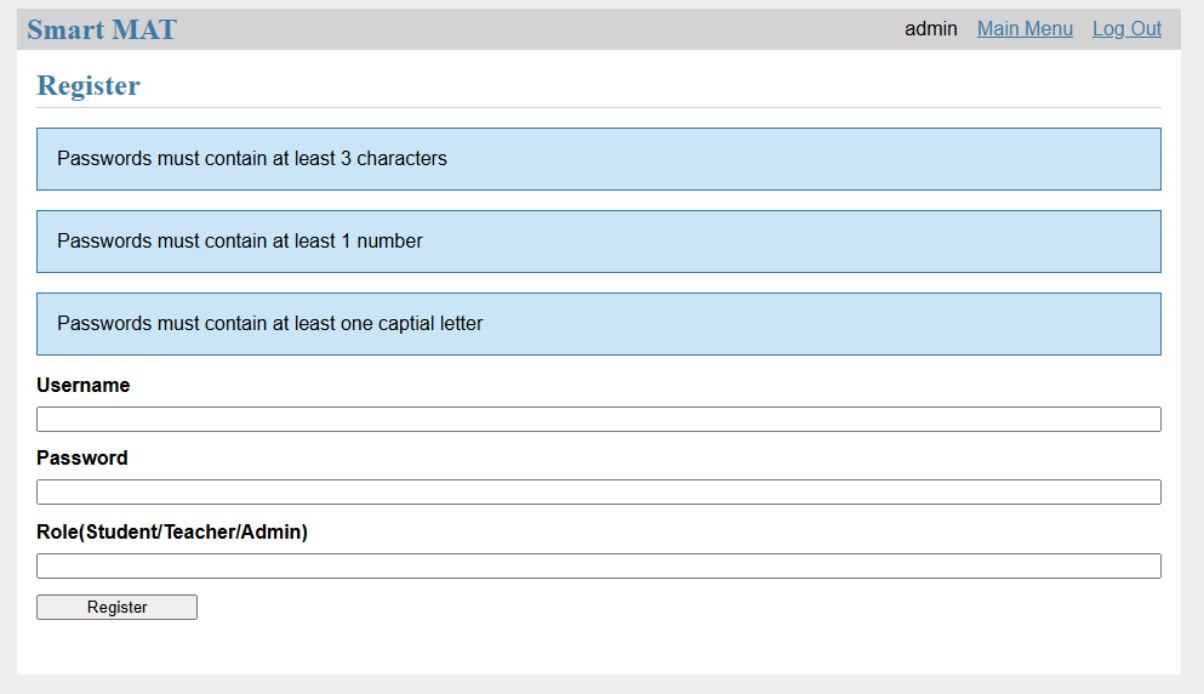
Admin users are able to register new users to the system.

Figure 17 - Register New Users



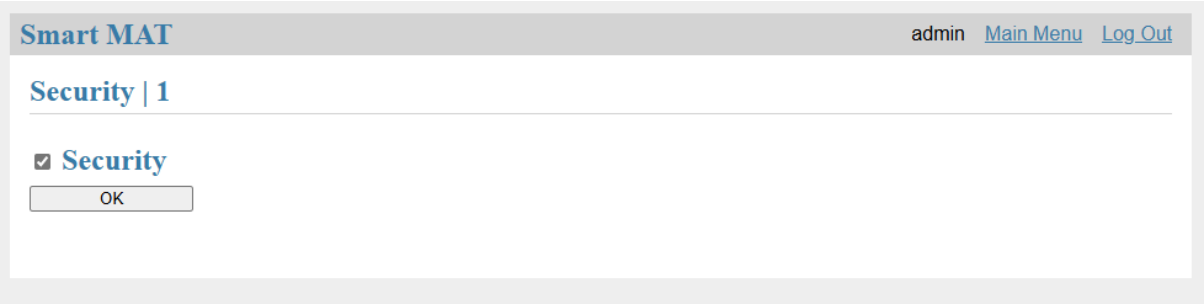
Further security ensures that passwords must match a Regex pattern.

Figure 18 - Regex Pattern



Admin users are also able to turn the security settings on or off.

Figure 19 - Security Settings



Application Programming Interface (API)

The system contains an Application Programming Interface (API). This can only be accessed by a user with Admin permissions. The API can be used to perform the following tasks:

Address	Task Description
127.0.0.1:5000/api/users/	List each user with their username and role
127.0.0.1:5000/api/users/?id=id Or 127.0.0.1:5000/api/user/id	List a specific user with the 'id' parameter
127.0.0.1:5000/api/users/?role=role Or 127.0.0.1:5000/api/user_r/role	List all users with a specified role using the 'role' parameter
127.0.0.1:5000/api/user_add/?username=uername&password=password&role=role/	Adds a user to the system with the username, password and role specified as arguments

127.0.0.1:5000/api/user_delete/?id=id/	Deletes a user from the system using the specified user id from the id argument

The examples below show how the API can be used

Figure 20 - API - List All Users

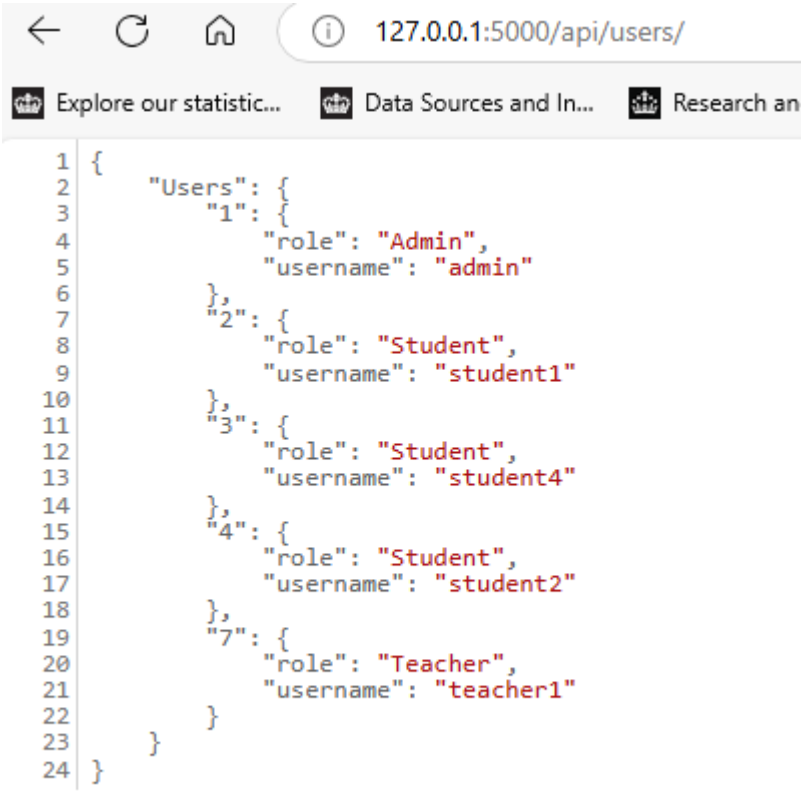


Figure 21- User with specific id

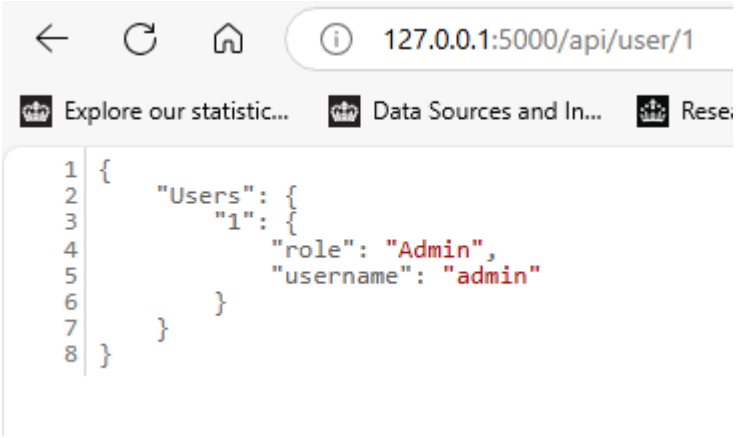
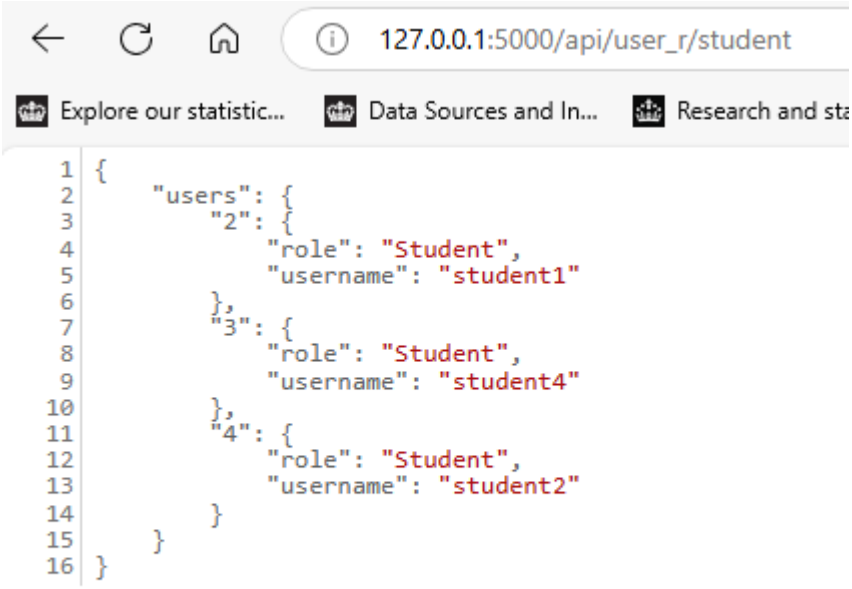


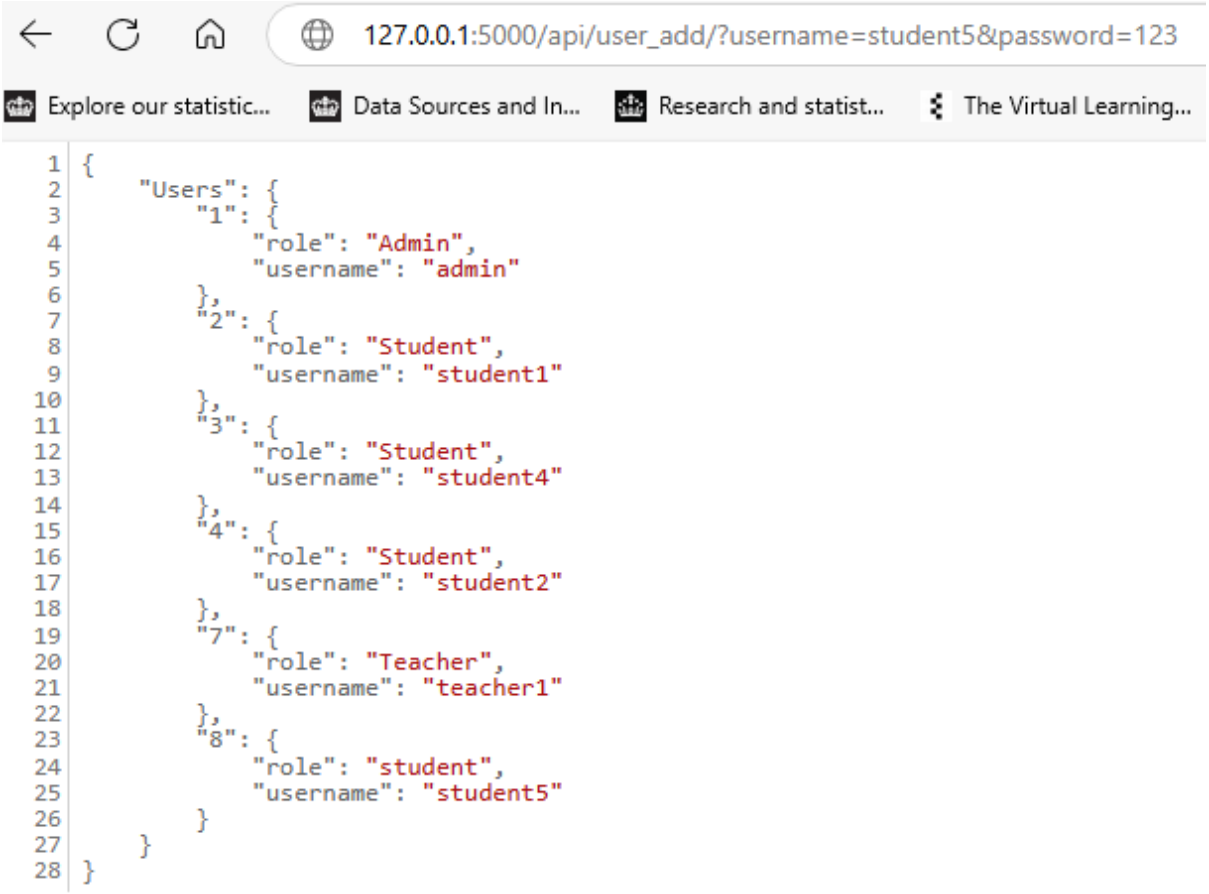
Figure 22 - users with a specific role



When a user is added to the system using the API, all users are displayed showing confirmation that the new user has been added. The example below has been generated using the API with:

http://127.0.0.1:5000/api/user_add/?username=student4&password=123

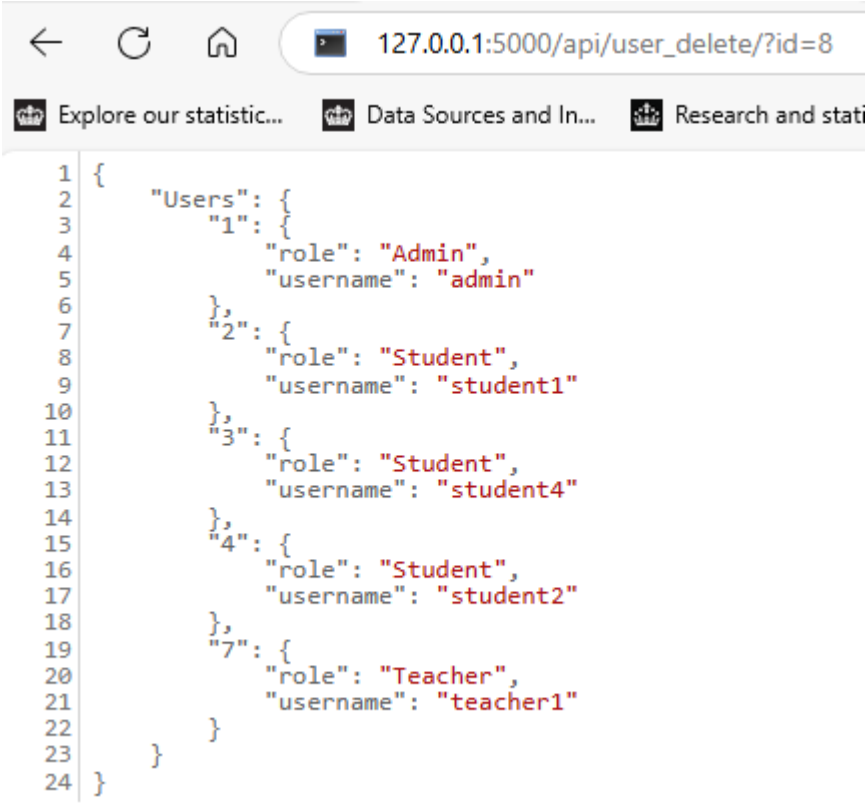
Figure 23 - Adding a user



Similarly, a user can be deleted. For example,

http://127.0.0.1:5000/api/user_delete/?id=8

Figure 24 - Deleting a user

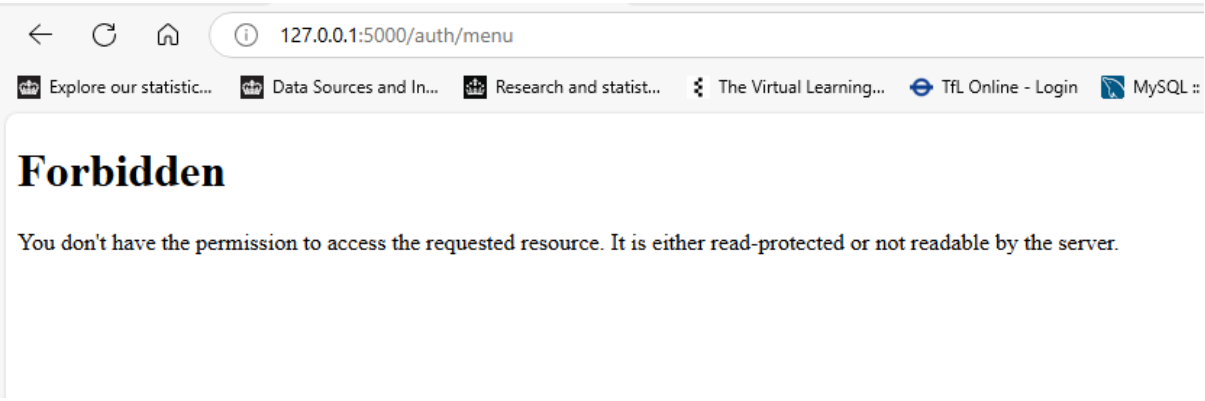


Preventing unauthorised access

It is not sufficient to limit the displays and options by role. Each page must be secured to prevent hackers simply typing the correct web-address to access part of the system. Throughout the application, there are checks for each page to ensure that unauthorised access is not granted. When a user tries to access a page beyond their allowed permissions the application returns the Forbidden error message. This is illustrated in the examples below:

Figure 25 - Access to the User List from a Student account

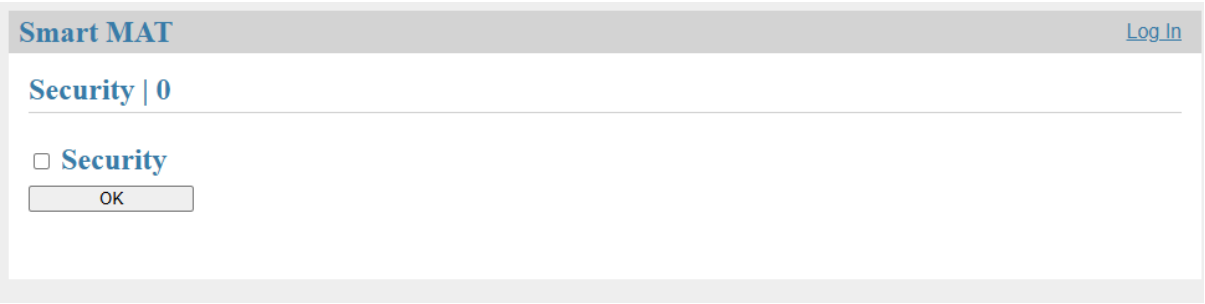
Figure 26 - Access to the menu when logged out



Using the application when security settings are turned off

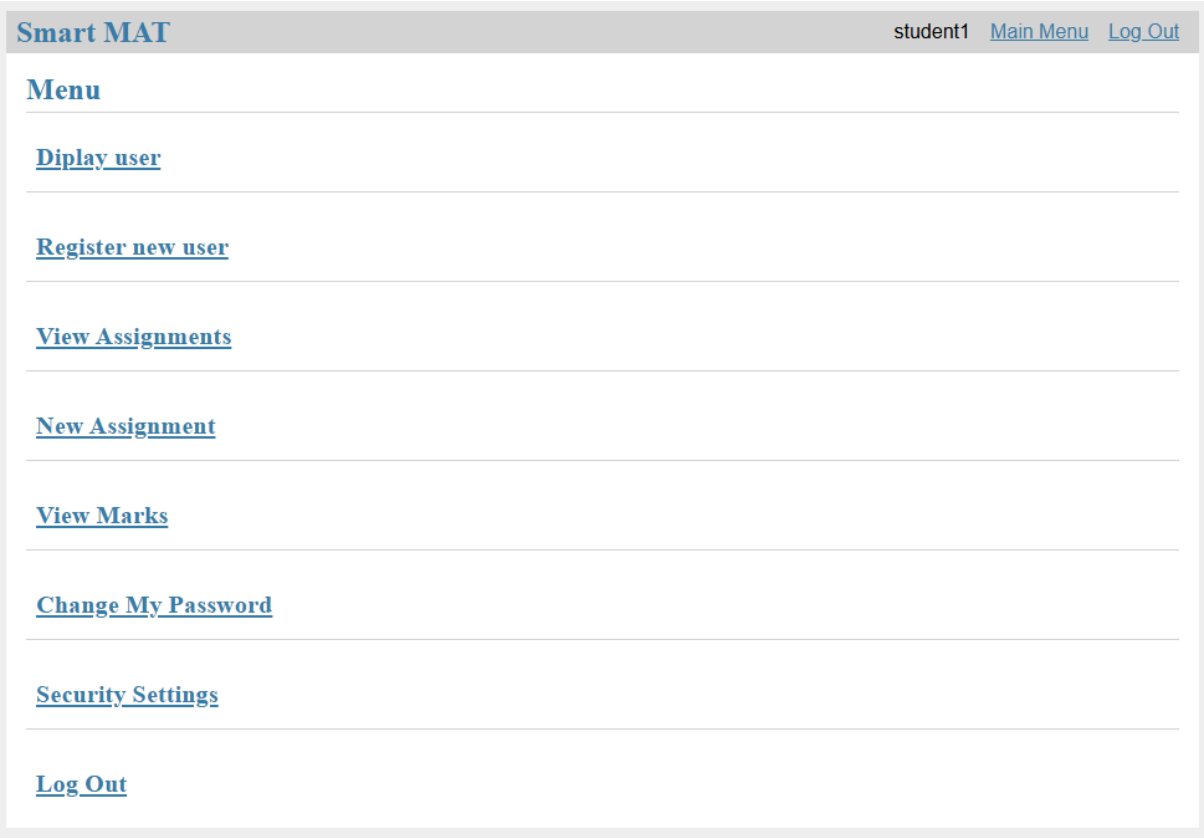
The application can be run without security

Figure 27 - Security Settings



When security settings are off, all users can access the full Admin menu and have access to all functionality regardless of their role. The example below shows a user with a Student role accessing the full menu, including Admin only options

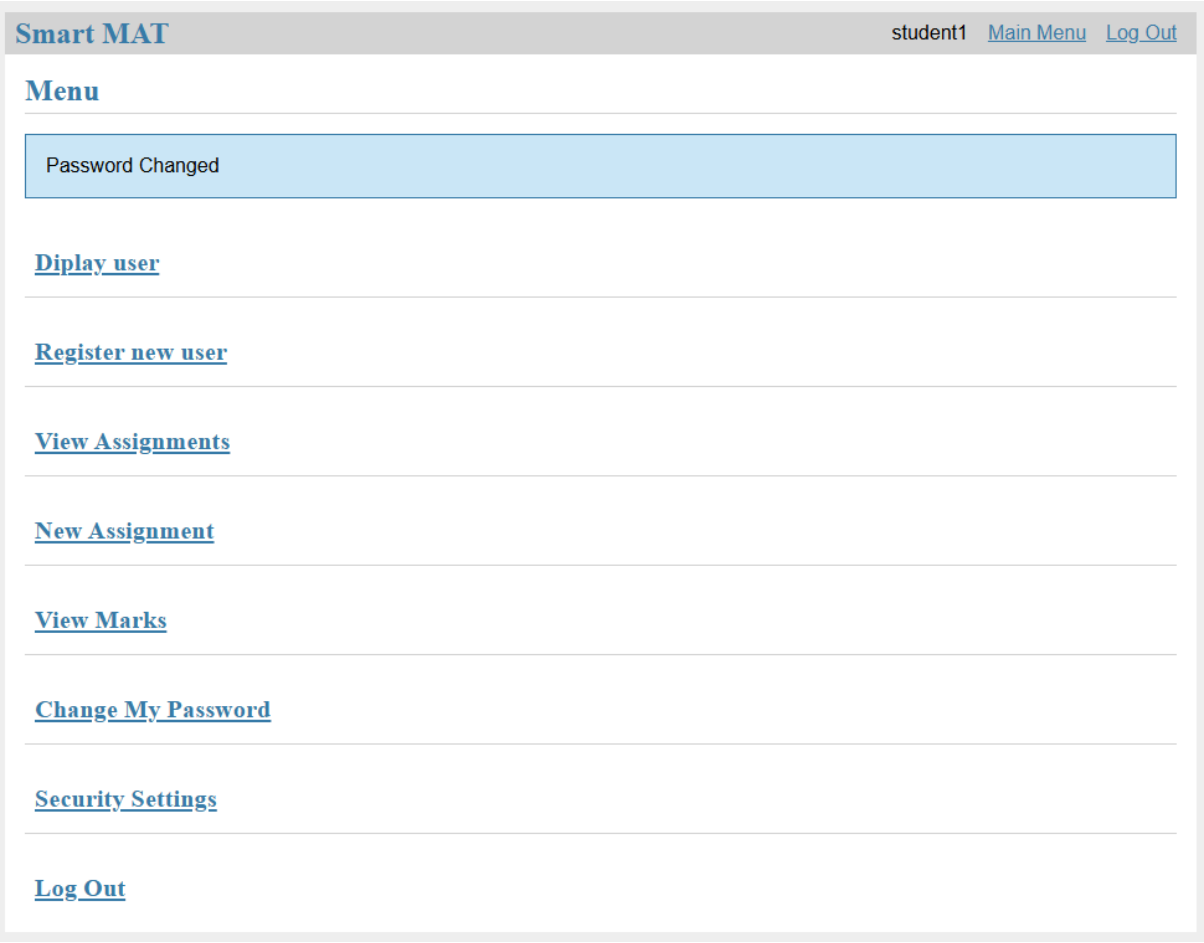
Figure 28 - Main menu with no security



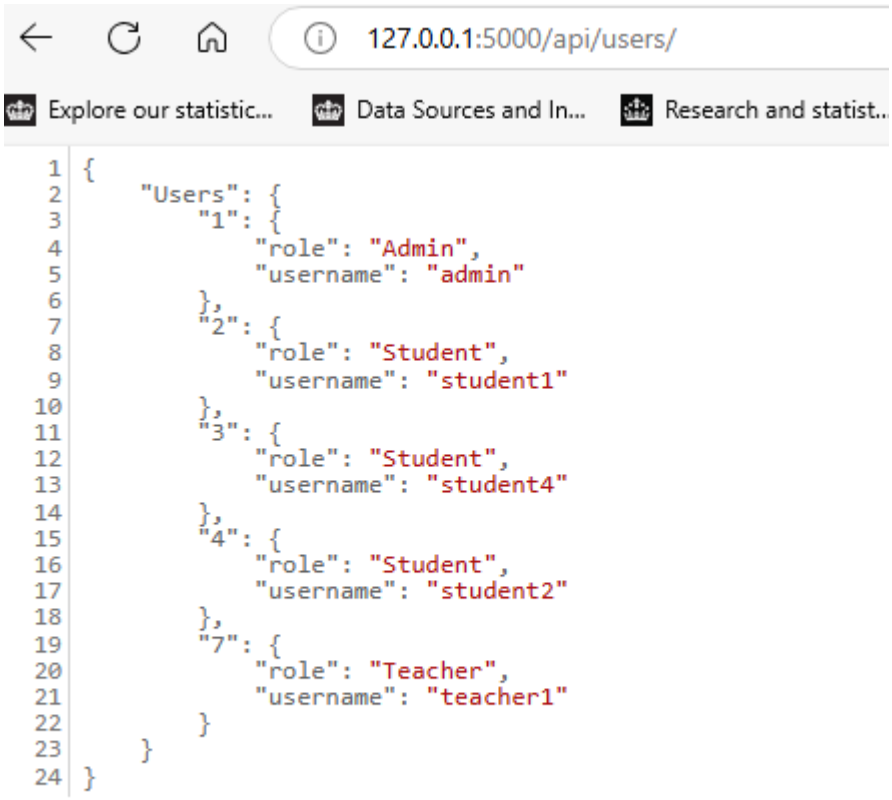
In addition to being able to access all functions within the system, the passwords can be changed, with no Regex pattern checks. The example below shows the password being successfully changed to '123' which would normally fail the Regex check.

Figure 29 - Password changed with no Regex check

Figure 30 - Returned to menu with a successful change of password

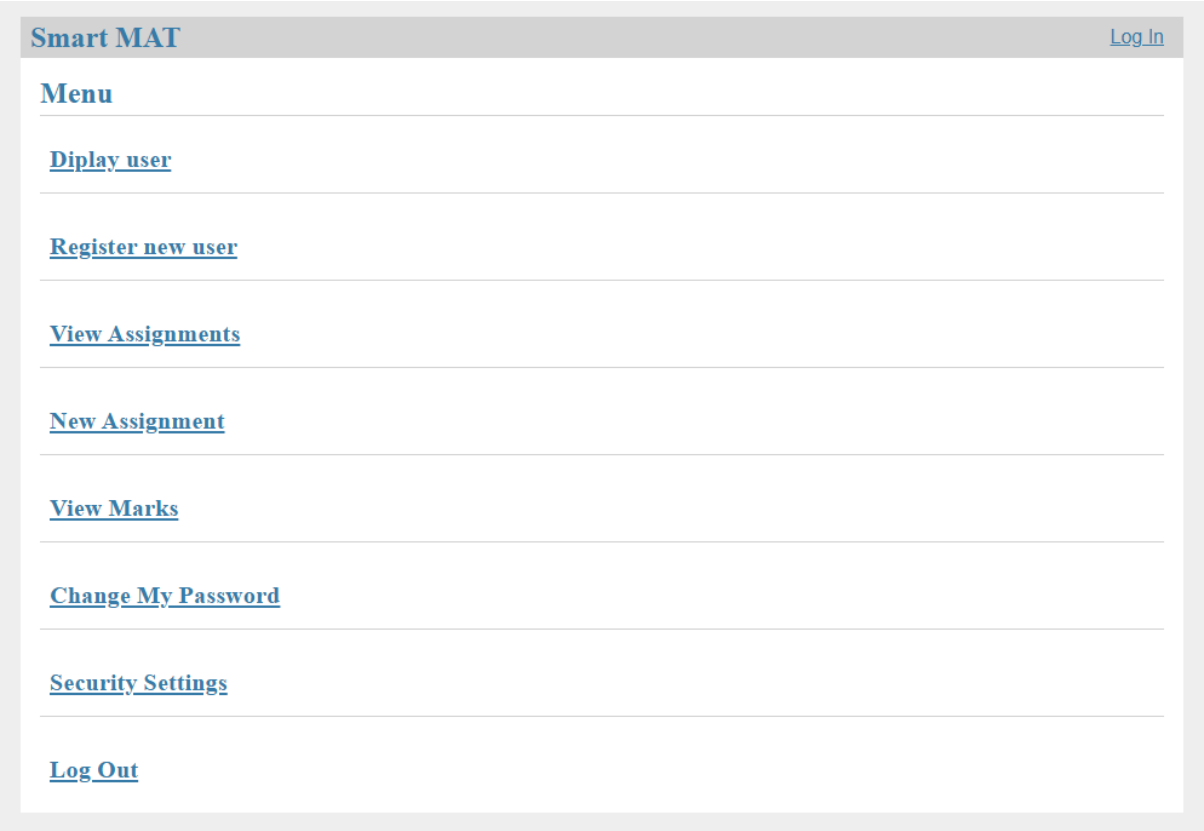


The API can also be accessed freely, by anyone with the knowledge of the web-address. The following example shows the results when security is turned off and the user is either logged in as a non-Admin user or the user is not logged into the application at all.



In addition, it is possible to access all of the functions, either directly or through the Admin menu when no user is logged in and the security settings are off. Note that no username is displayed and the option to ‘Log In’ is shown rather than ‘Log Out’

Figure 31 - Menu accessed when logged out and security is off



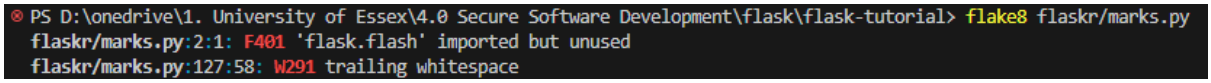
When the security is turned off it is possible for:

1. users' passwords to be changed by unauthorised users.
2. Users accounts or work to be deleted.
3. Students marks to be changed by unauthorised users.
4. Assignment posts to be deleted from the system.
5. New users to be created by unauthorised users.
6. The API to be accessed beyond those with an Admin role.
7. Full access to the system without being logged in.

Tests

Flake8 tests have been used to check the code against the best practices. The example below shows an unused module and a trailing whitespace.

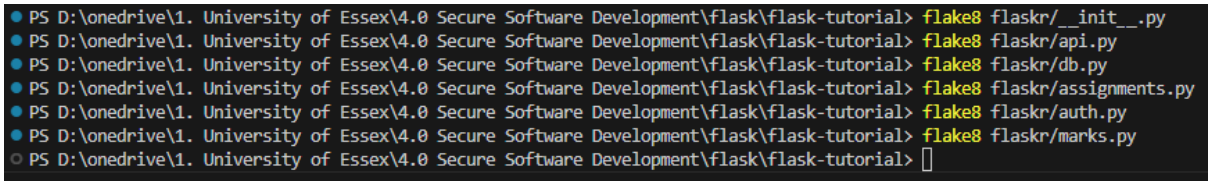
Figure 32 - Flake8 Testing



These were then corrected and Flake8 returned no response as there are no errors

Incorrect

Figure 33 - Flake8 Tests



Conclusion

The Se-LMS application, Smart MAT, meets the specifications of the design proposal. Specifically:

- Security can be turned on or off.
- Passwords meet a Regex pattern and are hashed.
- Teachers are able to set assignments, allocate them to students and provide feedback and marks.
- Students are able to view their assignments and submit marks.
- Role permissions are implemented throughout following the principle of least privilege.
- Login attempts are limited to prevent a brute force attack.
- Admin users can perform Create, Read, Update, and Delate (CRUD) functions via an API.
- GDPR principles are applied such that data is only shared where necessary.

References

Amazon Web Services, Inc. (2024). *What is an IDE? IDE Explained - AWS*. Available at: https://aws.amazon.com/what-is/ide/#:~:text=An%20integrated%20development%20environment%20(IDE [Accessed 28 August 2024]

Calder, A. (2018). *EU GDPR: A Pocket Guide, School's edition*. Cambridge: IT Governance Publishing. DOI: https://doi.org/10.2307/j.ctvvnfgq.

Chan, J. Chung, R. and Huang, J. (2019) Python API Development Fundamentals, Birmingham: Packt Publishing

Codeacdemy. (n.d.). *What is an IDE?* [online] Available at: https://www.codecademy.com/article/what-is-an-ide-ios [Accessed 28 August 2024]

Deligeorge, A. and Deligeorge, A. (2023). *Omnivision Design*. [online] OmnivisionDesign.com. Available at: Providing remote education: guidance for schools - GOV.UK (www.gov.uk) [Accessed 28 August 2024]

Department for Education (2024). Providing remote education: guidance for schools. Available from: https://www.gov.uk/government/publications/providing-remote-education-guidance-for-schools/providing-remote-education-guidance-for-schools [Accessed 09 September 2024].

Dewhurst, R. (2013). *Static Code Analysis | OWASP*. [online] Owasp.org. Available at: https://owasp.org/www-community/controls/Static_Code_Analysis [Accessed 28 August 2024]

Duckett, J. (2011). HTML & CSS: Design and Build Websites, Indianapolis: John Wiley & Sons.

Duckett, J. (2022). PHP & MYSL: Server-Side Web Development, New Jersey: John Wiley & Sons.

Farmaan, M. (2004). Secure Flask Sessions with Encryption: A Step-by-Step Guide. Available from: https://rb.gy/z2z9ga [Accessed: 26 August 2024].

Grinberg, M. (2014) Flask Web Development: Developing Web Applications With Python. Sebastopol: O'Reilly Media

Galluccio, E., Caselli, E. and Lombari, G. (2020). SQL Injection Strategies: Practical techniques to secure old vulnerabilities against modern attacks. Packt Publishing Ltd. DOI?

Kellezi, D., Boegelund, C. and Meng, W. (2021) Securing open banking with model-view-controller architecture and OWASP. *Wireless Communications and Mobile Computing*, (1): 1-13. DOI: https://doi.org/10.1155/2021/8028073.

Larson. E. (2018). [Research Paper] Automatic Checking of Regular Expressions. *IEEE 18th International Working Conference on Source Code Analysis*

and Manipulation (SCAM), Madrid, Spain, 2018. 225-234. DOI: https://doi.org/10.1109/SCAM.2018.00034.

MITRE. (2017). CWE VIEW: Weaknesses in OWASP Top Ten (2017). Available from: https://cwe.mitre.org/data/definitions/1026.html [Accessed 10 August 2024].

Ofsted (2013). Virtual learning environments: Eleven case studies of effective practice. Available from: https://assets.publishing.service.gov.uk/media/5a756171ed915d7314959844/VLE_e-portfolio_-_case_studies_booklet.pdf [Accessed 9 September 2024].

OSWAP (2021). Top 10 Web Application Security Risks. Available from: OWASP Top Ten | OWASP Foundation [Accessed 9 September 2024].

Pajankar, A. (2022). *Python Unit Test Automation*. Berkeley, CA: Apress. DOI: https://doi.org/10.1007/978-1-4842-7854-3_3

Patel, C., Gadhavi, M., & Patel, A. (2013). *A survey paper on e-learning based learning management Systems (LMS)*. *International Journal of Scientific & Engineering Research (IJSER)*, **4**(6), 171.

Pillai, A.B. (2017). Software Architecture with Python. 1st ed. Birmingham: Packt Publishing.

Soumya (2019). *Version Control Systems*. [online] Geeks for Geeks. Available from: https://www.geeksforgeeks.org/version-control-systems/[Accessed 28 Aug. 2024]

Spraul, V.A. (2015). How Software Works: The Magic Behind Encryption, CGI, Search Engines, and Other Everyday Technologies. San Francisco: No Starch Press.