

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

Figure 1 - Login.....	2
Figure 2 - Incorrect Login	3
Figure 3 – Admin Menu	5
Figure 4 - Teacher Menu	5
Figure 5 - Student Menu	6
Figure 6 - View Student Assignment.....	6
Figure 7 - Student View Marks & Submit Work.....	6
Figure 8 - Submit Assignment.....	7
Figure 9 - Updated Submission.....	7
Figure 10 - View Assignments - Teacher	8
Figure 11 - Edit Assignment.....	8
Figure 12 - Assignment Marks - Teacher	9
Figure 13 - Assign users	9
Figure 14 - Teacher Marks	9
Figure 15 - Edit Marks	10
Figure 16 - Display and Edit Users.....	10
Figure 17 - Register New Users	11
Figure 18 - Regex Pattern	11

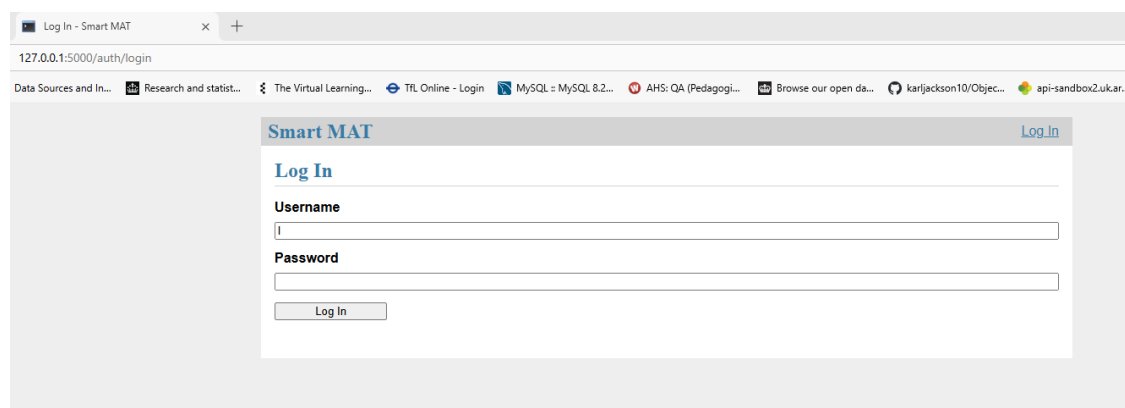
Figure 19 - Security Settings	12
Figure 20 - API - List All Users	13
Figure 21- User with specific id.....	14
Figure 22 - users with a specific role	14
Figure 23 - Adding a user.....	15
Figure 24 - Deleting a user.....	16
Figure 25 - Access to the User List from a Student account	17
Figure 26 - Access to the menu when logged out	17
Figure 27 - Security Settings	17
Figure 28 - Main menu with no security	18
Figure 29 - Password changed with no Regex check	18
Figure 30 - Returned to menu with a successful change of password	19
Figure 31 - Menu accessed when logged out and security is off.....	21
Figure 32 - Flake8 Testing	22
Figure 33 - Flake8 Tests.....	22

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 screenshot shows a web browser window with the title 'Log In - Smart MAT'. The address bar displays '127.0.0.1:5000/auth/login'. The browser's tab bar shows several open tabs, including 'Data Sources and In...', 'Research and statist...', 'The Virtual Learning...', 'TIL Online - Login', 'MySQL: MySQL 8.2...', 'AHS: QA (Pedagogi...', 'Browse our open da...', 'karljackson10/Objec...', and 'api-sandbox2.uk.ar...'. The main content area of the browser shows the 'Smart MAT' application. At the top right of the application is a 'Log In' link. Below this is a 'Log In' section with two input fields: 'Username' and 'Password'. The 'Username' field contains the letter 'i'. Below the 'Password' field is a 'Log In' button.

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

The figure consists of two screenshots of the Smart MAT login interface, illustrating the state after an incorrect password attempt.

Top Screenshot (1st Attempt):

- Header:** "Smart MAT" on the left and a "[Log In](#)" link on the right.
- Title:** "Log In" in blue text.
- Message:** A light blue box displays "Incorrect password. 1 Incorect attempts".
- Fields:** "Username" and "Password" labels above their respective input fields.
- Button:** A "Log In" button at the bottom.

Bottom Screenshot (2nd Attempt):

- Header:** "Smart MAT" on the left and a "[Log In](#)" link on the right.
- Title:** "Log In" in blue text.
- Message:** A light blue box displays "Incorrect password. 2 Incorect attempts".
- Fields:** "Username" and "Password" labels above their respective input fields.
- Button:** A "Log In" button at the bottom.

Smart MAT

[Log In](#)

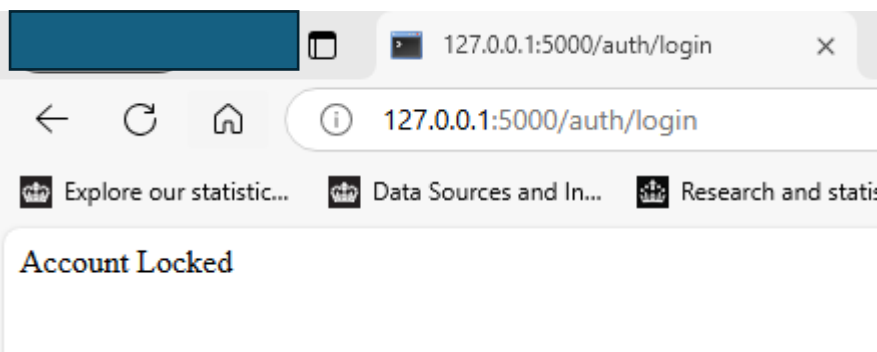
Log In

Incorrect password. 3 Incorect attempts

Username

Password

Log In



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

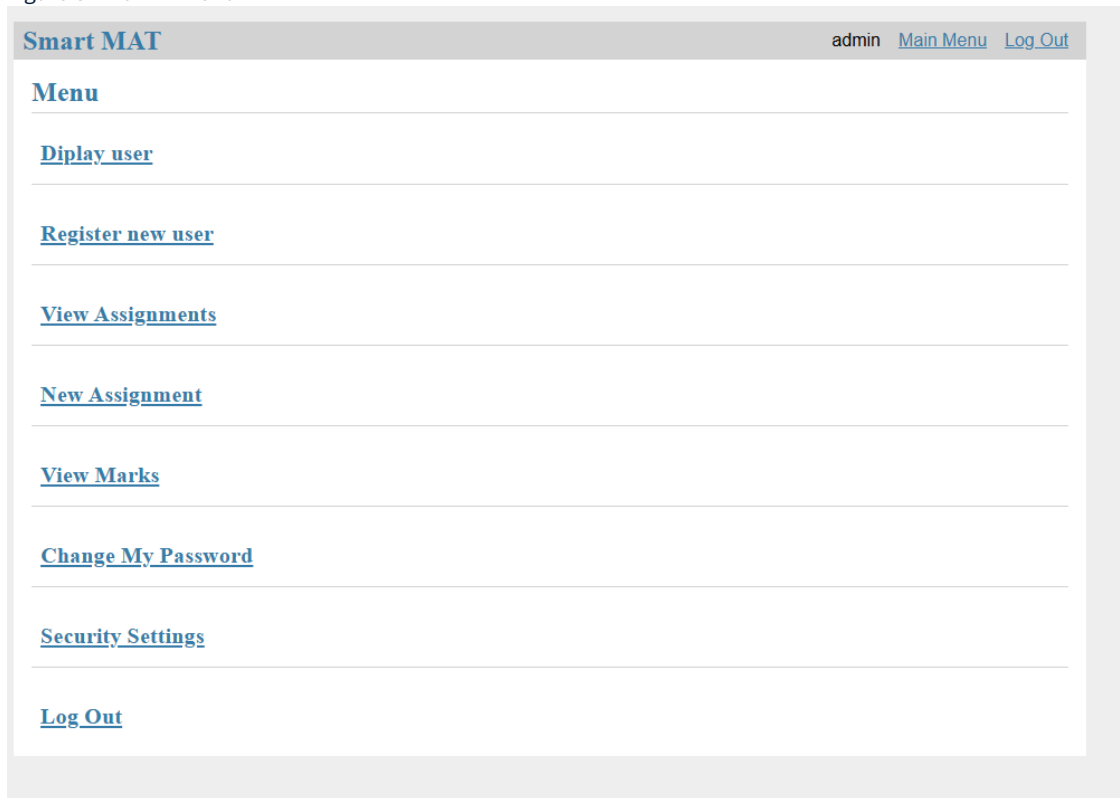


Figure 4 - Teacher Menu



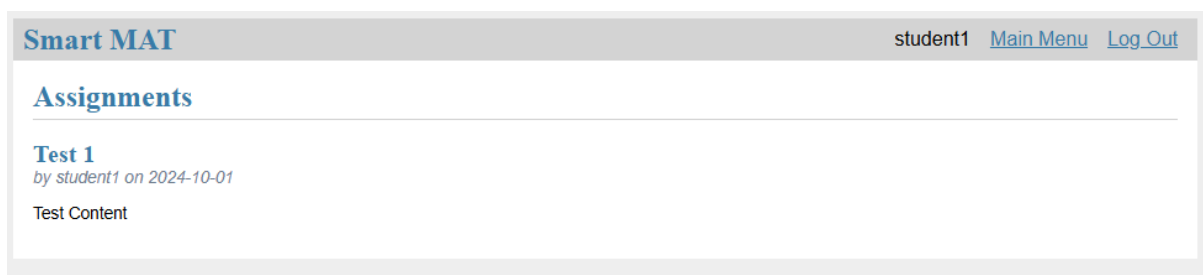
Figure 5 - Student Menu



Functions by role: Students

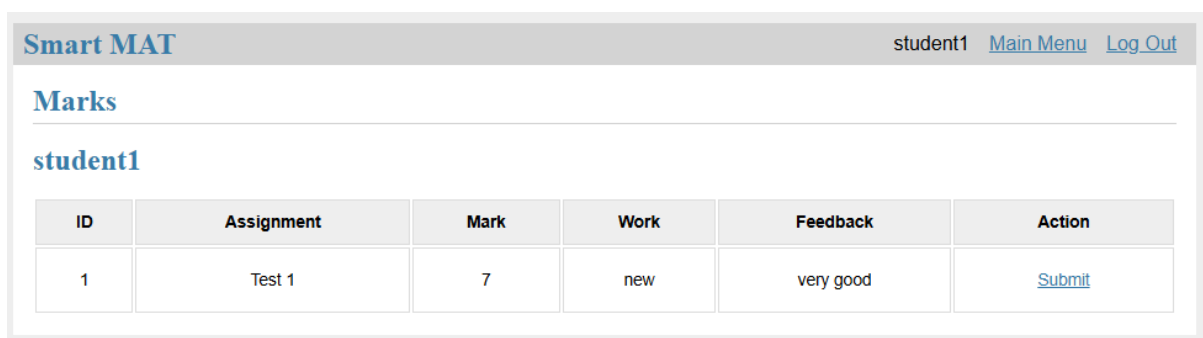
Student access is restricted to the functions necessary for their role. Students can only see assignments that have been allocated to them

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

Smart MAT

student1 [Main Menu](#) [Log Out](#)

Submit Assignment | Test 1

Submission

new work

Save

Figure 9 - Updated Submission

Smart MAT

student1 [Main Menu](#) [Log Out](#)

Marks

student1

ID	Assignment	Mark	Work	Feedback	Action
1	Test 1	7	new work	very good	Submit

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

Smart MAT

student1 [Main Menu](#) [Log Out](#)

Change Password | student1

Passwords must contain at least 3 characters

Passwords must contain at least one captial letter

Passwords must contain at least one lower case letter

Password

1

Save

Functions by role: Teachers

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

Figure 10 - View Assignments - Teacher

Smart MAT

teacher1Main MenuLog Out

Assignments

New

Test 2

by teacher1 on 2024-10-14

Test

Edit

Test 1

by admin on 2024-10-01

Test Content

Teachers can edit or delete their own assignments.

Figure 11 - Edit Assignment

Smart MAT

teacher1Main MenuLog Out

Edit "Test 2"

Title

Test 2

Body

Test

Save

Delete

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

Figure 12 - Assignment Marks - Teacher

Smart MAT

teacher1Main MenuLog Out

Marks

1 | Test 1

by admin on 2024-10-01

2 | Test 2

by teacher1 on 2024-10-14

AssignView

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

Figure 13 - Assign users

Smart MAT

teacher1Main MenuLog Out

Assign

Test 2

View

ID	Student	Mark	Feedback	Assigned	Assign	Remove
1	admin	None	None		<input type="checkbox"/>	
2	student1	None	None	✓		<input type="checkbox"/>
3	student4	None	None		<input type="checkbox"/>	
4	student2	None	None	✓		<input type="checkbox"/>
7	teacher1	None	None		<input type="checkbox"/>	

They can also view the marks:

Figure 14 - Teacher Marks

Smart MAT

teacher1Main MenuLog Out

Marks

Test 2 | 2

Assign

ID	Student	Mark	Work	Feedback	Action
2	student1	None	None	None	Edit
4	student2	2	None	Try Harder	Edit

Teachers can edit the marks and feedback, for their assignments

Figure 15 - Edit Marks

Smart MAT

teacher1 [Main Menu](#) [Log Out](#)

Edit | Test 2

Mark

Feedback

Try Harder

Save

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

Smart MAT

admin [Main Menu](#) [Log Out](#)

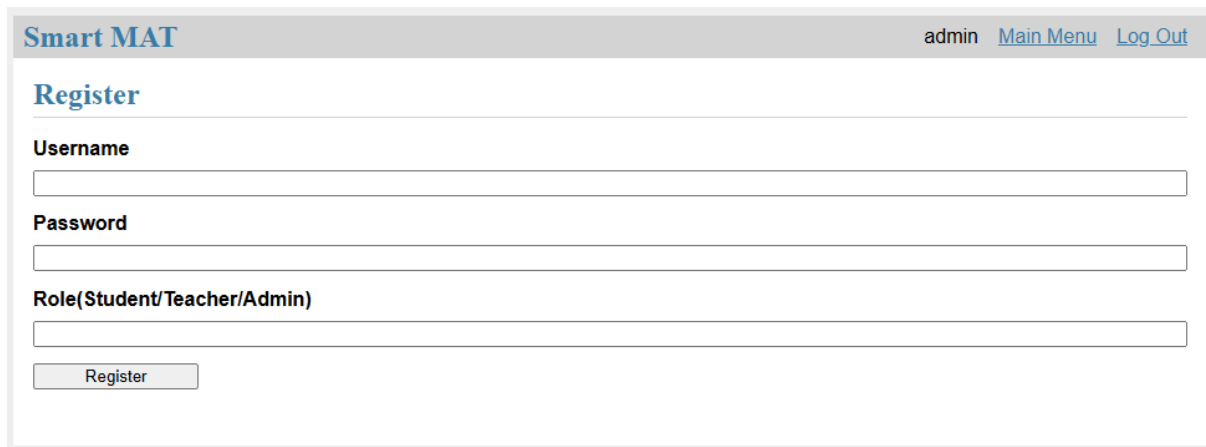
Users

[New User](#)

1 admin	Role: Admin	Edit	<div>Delete</div>
2 student1	Role: Student	Edit	<div>Delete</div>
3 student4	Role: Student	Edit	<div>Delete</div>
4 student2	Role: Student	Edit	<div>Delete</div>
7 teacher1	Role: Teacher	Edit	<div>Delete</div>

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

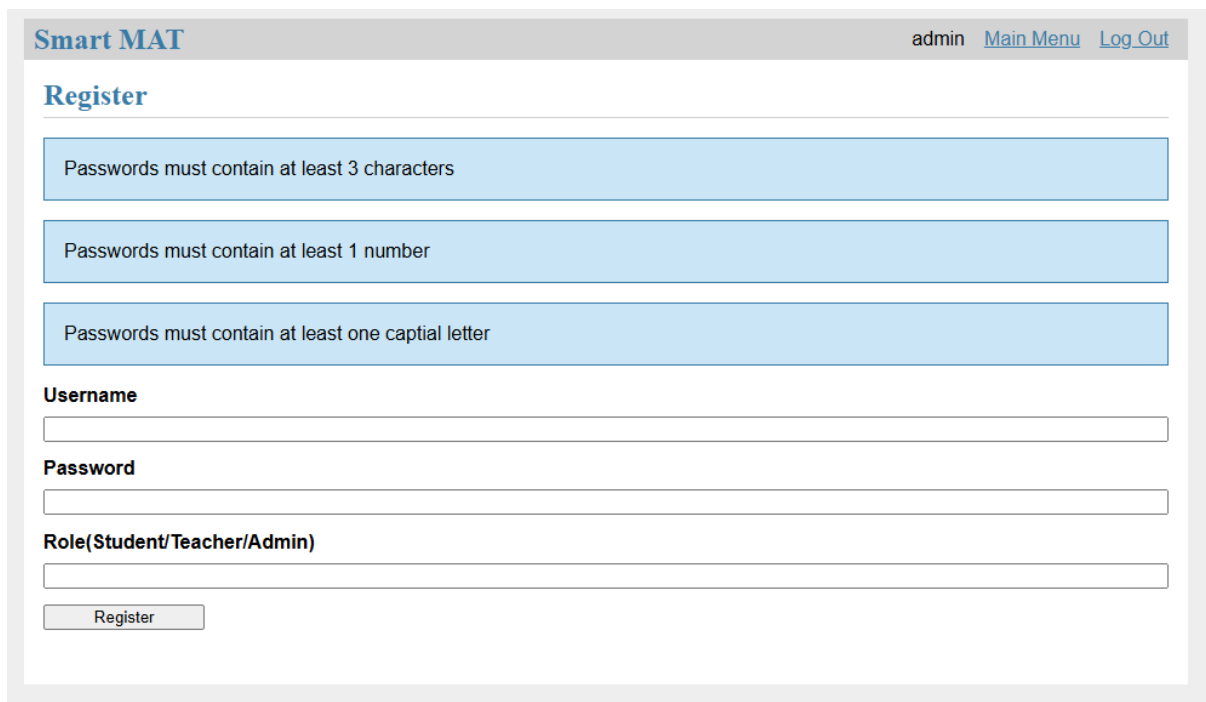
Figure 17 - Register New Users



The image shows a web form titled "Smart MAT" with a header bar containing "admin", "Main Menu", and "Log Out" links. The form is titled "Register" and contains three input fields: "Username", "Password", and "Role(Student/Teacher/Admin)". Below the fields is a "Register" button.

Further security ensures that passwords must match a Regex pattern.

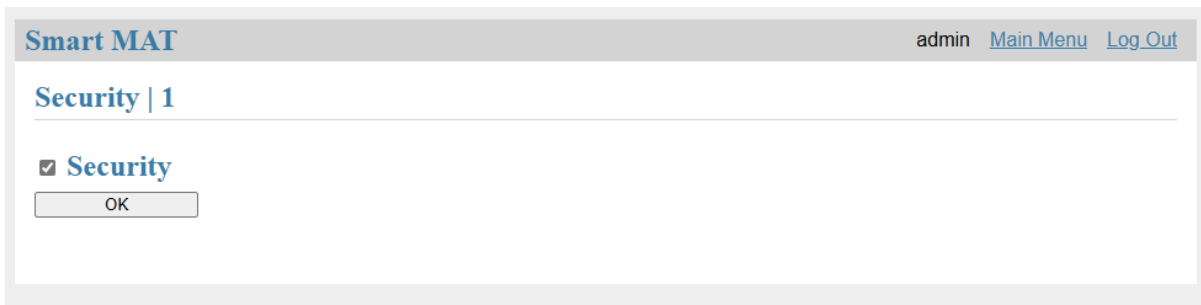
Figure 18 - Regex Pattern



The image shows the same "Smart MAT" web form as Figure 17, but with additional security requirements for passwords. Above the "Password" field, there are three blue boxes containing the following text: "Passwords must contain at least 3 characters", "Passwords must contain at least 1 number", and "Passwords must contain at least one captial letter". The form also includes the "Username", "Role(Student/Teacher/Admin)", and "Register" button fields.

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

Figure 19 - Security Settings



Smart MAT admin Main Menu Log Out

Security | 1

☒ Security

OK

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=username&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

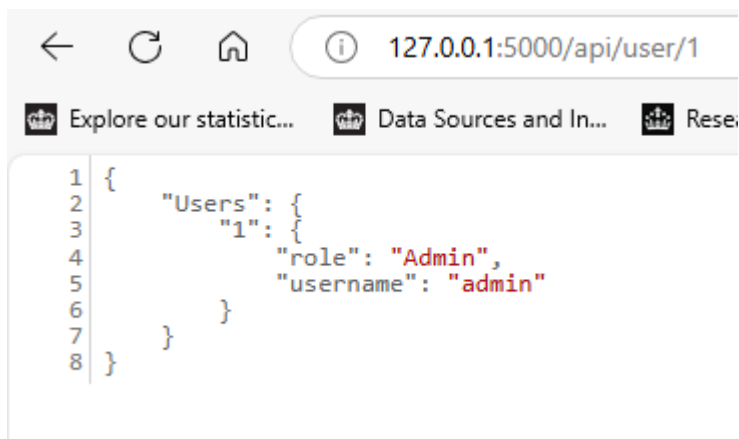
The screenshot shows a web browser with the address bar displaying `127.0.0.1:5000/api/users/`. Below the address bar, there are three tabs: "Explore our statistic...", "Data Sources and In...", and "Research an...". The main content area displays a JSON response from the API, which is a list of users. The JSON is formatted with line numbers on the left side, ranging from 1 to 24. The response is a JSON object with a key "Users" that points to an array of user objects. Each user object contains a unique ID, a role, and a username.

```

1 {
2   "Users": {
3     "1": {
4       "role": "Admin",
5       "username": "admin"
6     },
7     "2": {
8       "role": "Student",
9       "username": "student1"
10    },
11    "3": {
12      "role": "Student",
13      "username": "student4"
14    },
15    "4": {
16      "role": "Student",
17      "username": "student2"
18    },
19    "7": {
20      "role": "Teacher",
21      "username": "teacher1"
22    }
23  }
24 }

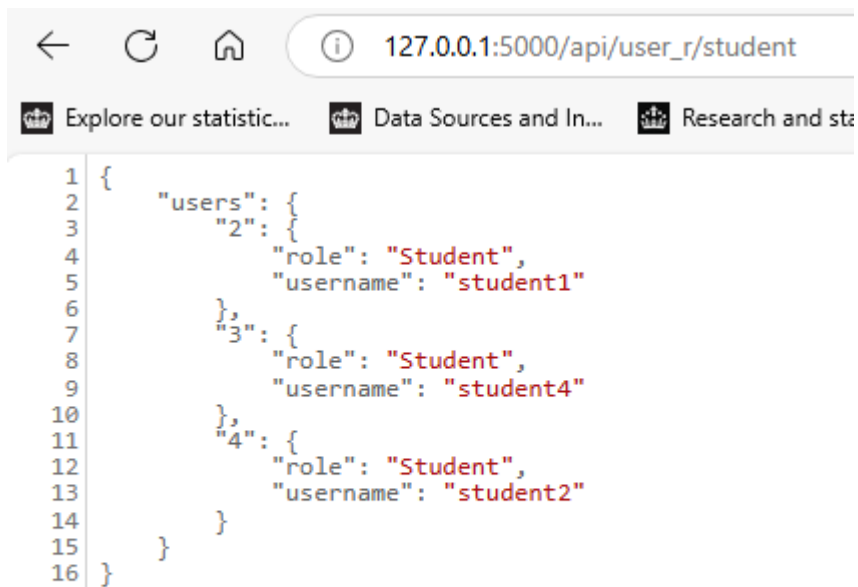
```

Figure 21- User with specific id



```
1 {
2   "Users": {
3     "1": {
4       "role": "Admin",
5       "username": "admin"
6     }
7   }
8 }
```

Figure 22 - users with a specific role

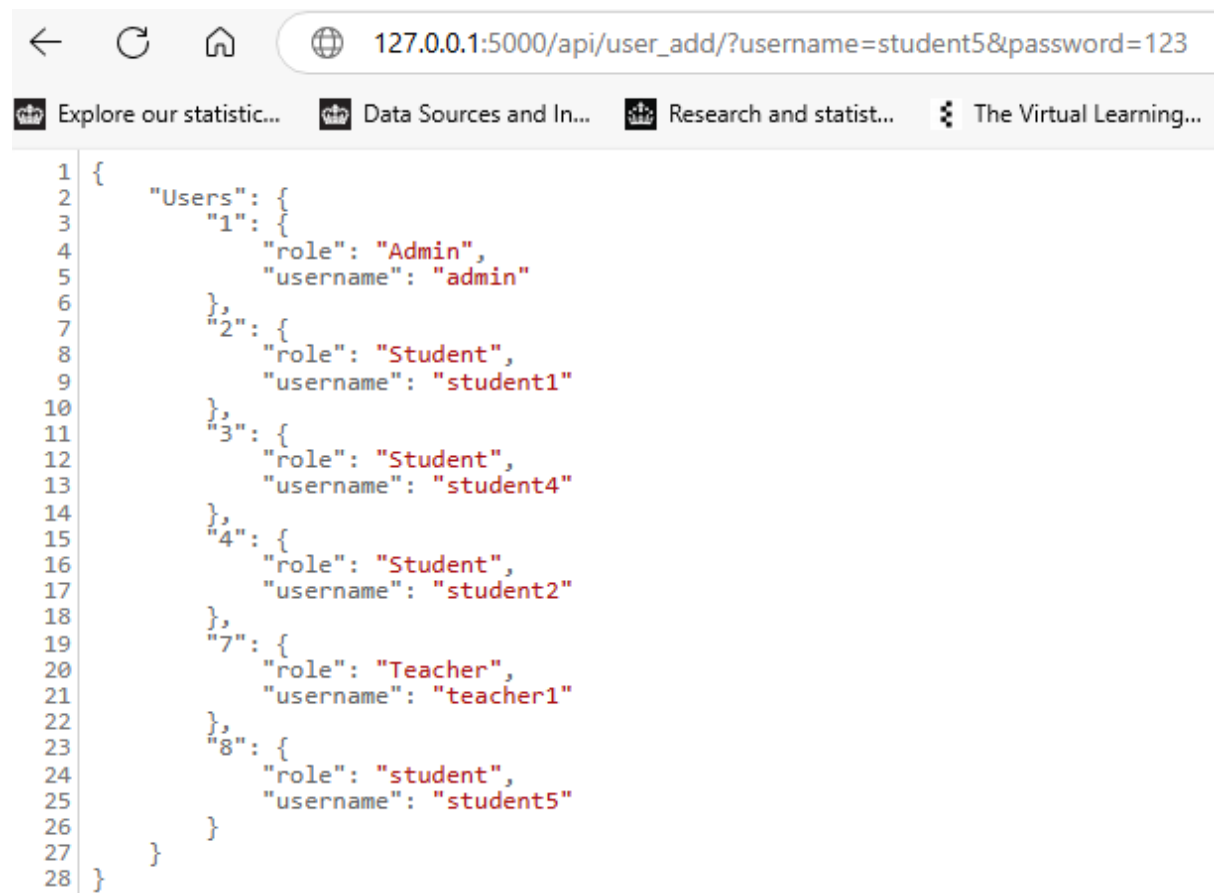


```
1 {
2   "users": {
3     "2": {
4       "role": "Student",
5       "username": "student1"
6     },
7     "3": {
8       "role": "Student",
9       "username": "student4"
10    },
11    "4": {
12      "role": "Student",
13      "username": "student2"
14    }
15  }
16 }
```

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



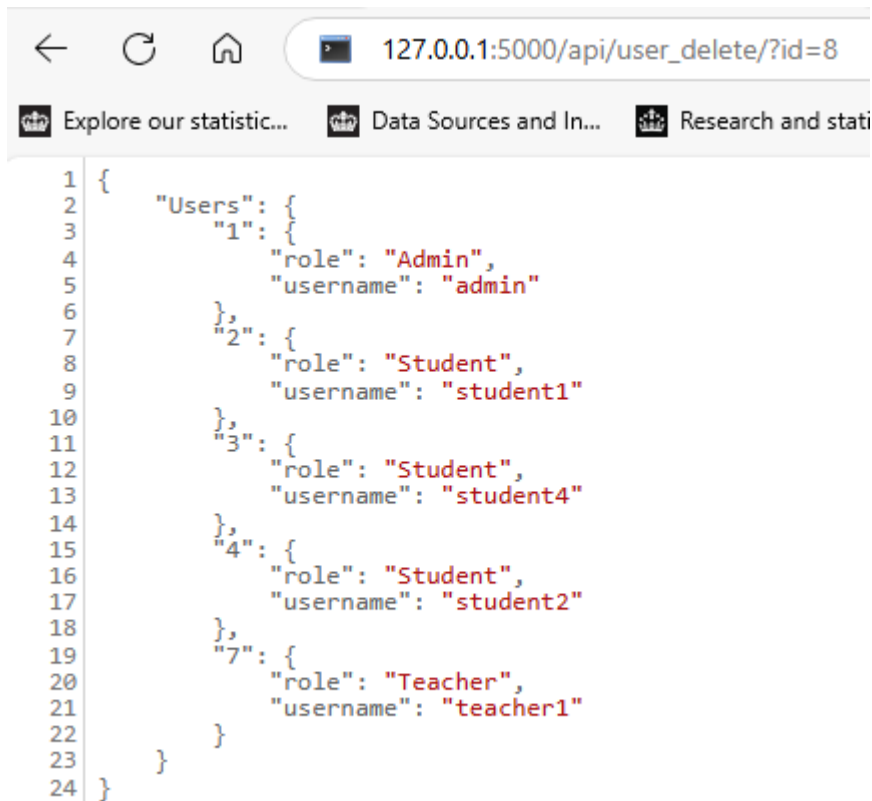
The screenshot shows a web browser window with the address bar displaying the URL: `127.0.0.1:5000/api/user_add/?username=student5&password=123`. Below the address bar, there are four tabs: "Explore our statistic...", "Data Sources and In...", "Research and statist...", and "The Virtual Learning...". The main content area displays a JSON response from the API, which is a list of users. The JSON is formatted with line numbers on the left side, ranging from 1 to 28. The response is a JSON array containing a "Users" object, which is a list of user objects. Each user object contains "role" and "username" fields. The users listed are: an Admin user with username "admin", and several Student users with usernames "student1", "student4", "student2", and "student5". There is also a Teacher user with username "teacher1".

```
1 {
2   "Users": {
3     "1": {
4       "role": "Admin",
5       "username": "admin"
6     },
7     "2": {
8       "role": "Student",
9       "username": "student1"
10    },
11    "3": {
12      "role": "Student",
13      "username": "student4"
14    },
15    "4": {
16      "role": "Student",
17      "username": "student2"
18    },
19    "7": {
20      "role": "Teacher",
21      "username": "teacher1"
22    },
23    "8": {
24      "role": "student",
25      "username": "student5"
26    }
27  }
28 }
```

Similarly, a user can be deleted. For example,

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

Figure 24 - Deleting a user



The screenshot shows a web browser window with a REST client interface. The address bar displays the URL `127.0.0.1:5000/api/user_delete/?id=8`. Below the address bar, there are three tabs: "Explore our statistic...", "Data Sources and In...", and "Research and stati". The main content area shows a JSON response from the API, which is a list of users. The JSON is formatted with line numbers 1 through 24 on the left. The response is a JSON object with a "Users" key, which contains a list of user objects. Each user object has a "role" and a "username" property. The roles are "Admin", "Student", and "Teacher". The usernames are "admin", "student1", "student4", "student2", and "teacher1".

```
1 {  
2   "Users": {  
3     "1": {  
4       "role": "Admin",  
5       "username": "admin"  
6     },  
7     "2": {  
8       "role": "Student",  
9       "username": "student1"  
10    },  
11    "3": {  
12      "role": "Student",  
13      "username": "student4"  
14    },  
15    "4": {  
16      "role": "Student",  
17      "username": "student2"  
18    },  
19    "7": {  
20      "role": "Teacher",  
21      "username": "teacher1"  
22    }  
23  }  
24 }
```

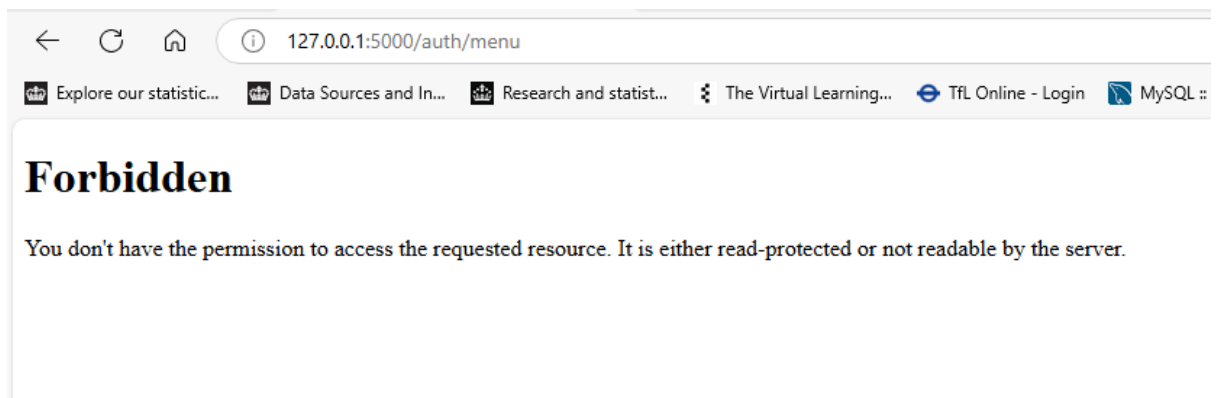
Preventing unauthorised access

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



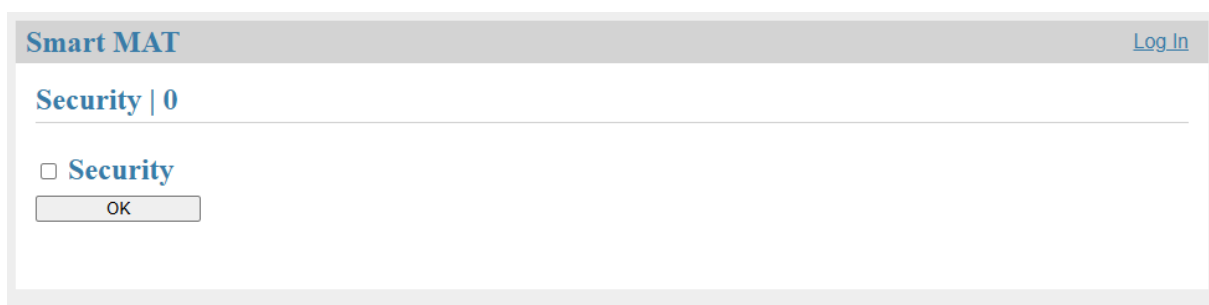
Figure 26 - Access to the menu when logged out



Using the application when security settings are turned off

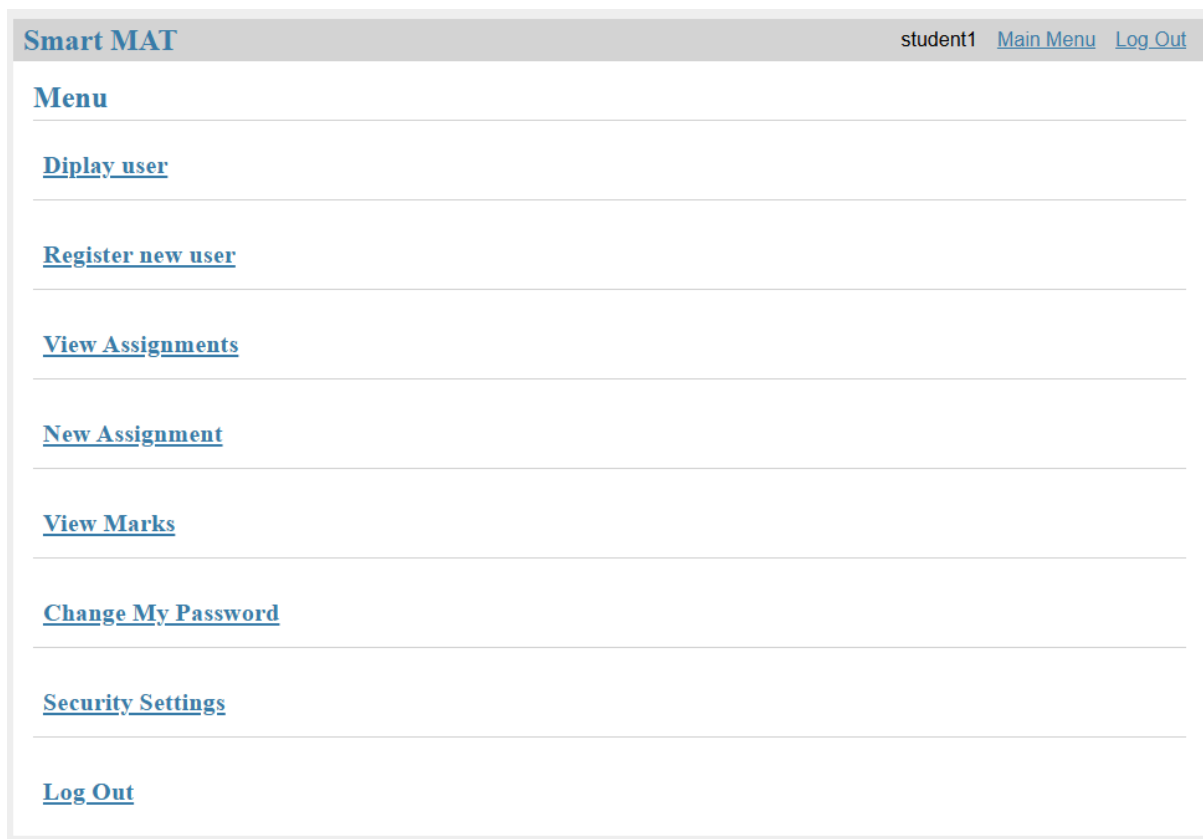
The application can be run without security

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



The screenshot shows the 'Smart MAT' application interface. At the top, a header bar contains the application name 'Smart MAT' on the left and the user 'student1' with links to 'Main Menu' and 'Log Out' on the right. Below the header, the page is titled 'Menu'. A list of menu items is displayed, each underlined and separated by a horizontal line: 'Diplay user', 'Register new user', 'View Assignments', 'New Assignment', 'View Marks', 'Change My Password', 'Security Settings', and 'Log Out'.

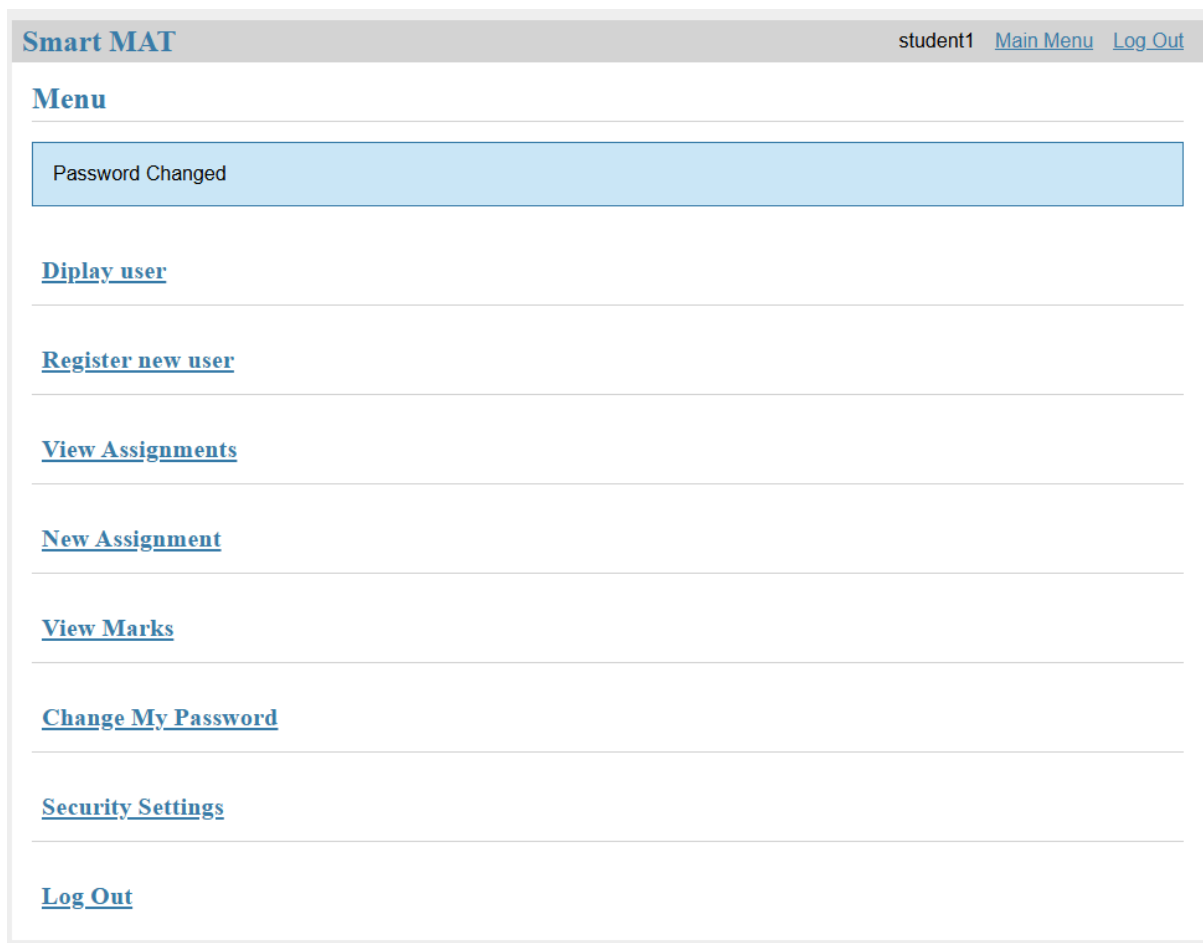
In addition to being able to access all functions within the system, the passwords can be changed, with no Regex pattern checks. The example below shows the password being successfully changed to '123' which would normally fail the Regex check.

Figure 29 - Password changed with no Regex check

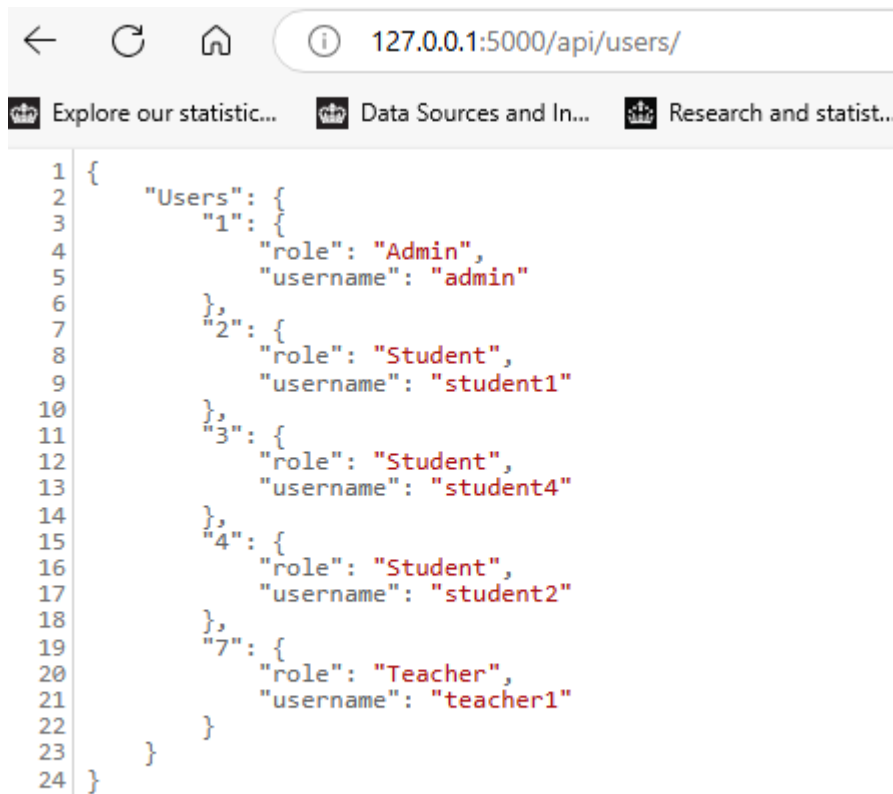


The screenshot shows the 'Change Password | admin' form. The header bar is identical to the previous figure. The page title is 'Change Password | admin'. Below the title, there is a label 'Password' followed by a text input field containing the value '123'. A 'Save' button is located below the input field.

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.

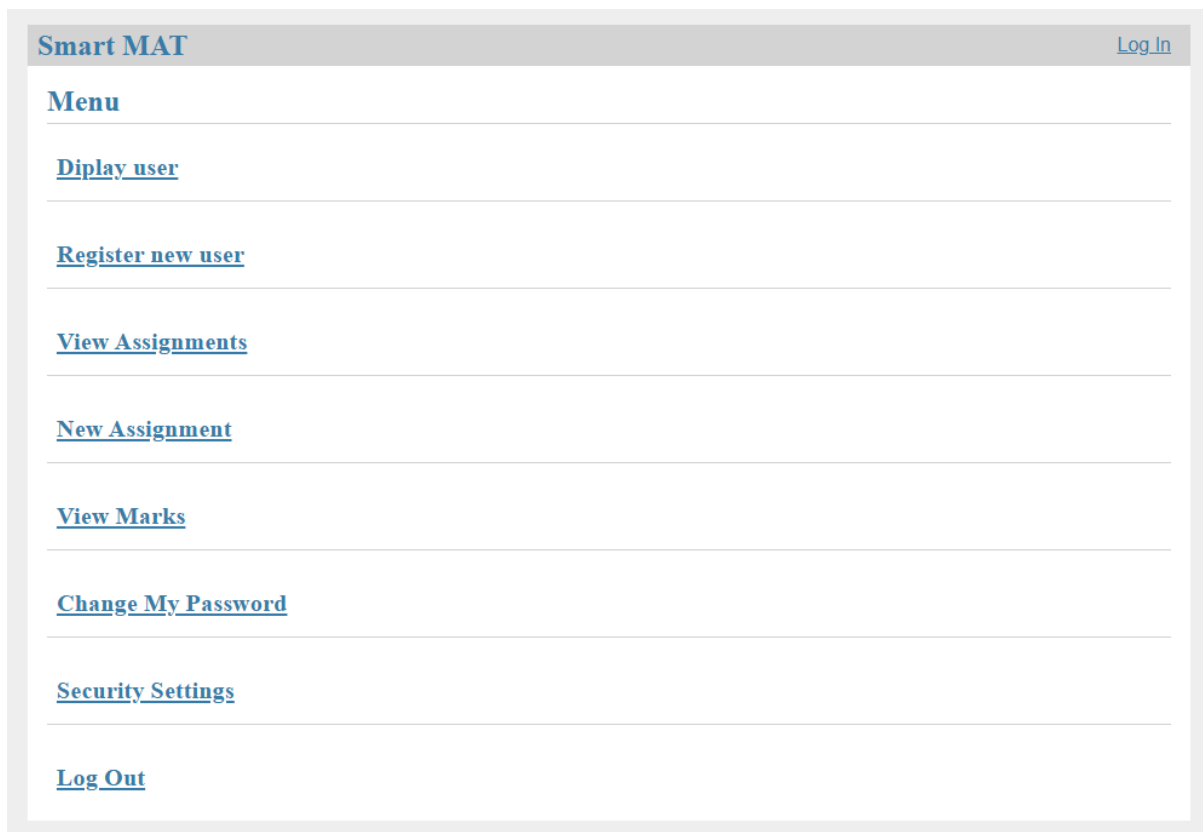


The screenshot shows a web browser window with the address bar displaying `127.0.0.1:5000/api/users/`. Below the address bar, there are three tabs: "Explore our statistic...", "Data Sources and In...", and "Research and statist..". The main content area displays a JSON response from the API, which is a list of users. The JSON is formatted with line numbers on the left side, ranging from 1 to 24. The response is a JSON object with a key "Users" that points to an array of user objects. Each user object contains a "role" and a "username". The roles are "Admin", "Student", and "Teacher". The usernames are "admin", "student1", "student4", "student2", and "teacher1".

```
1 {
2   "Users": {
3     "1": {
4       "role": "Admin",
5       "username": "admin"
6     },
7     "2": {
8       "role": "Student",
9       "username": "student1"
10    },
11    "3": {
12      "role": "Student",
13      "username": "student4"
14    },
15    "4": {
16      "role": "Student",
17      "username": "student2"
18    },
19    "7": {
20      "role": "Teacher",
21      "username": "teacher1"
22    }
23  }
24 }
```

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

```
PS D:\onedrive\1. University of Essex\4.0 Secure Software Development\flask\flask-tutorial> flake8 flaskr/marks.py
flaskr/marks.py:2:1: F401 'flask.flash' imported but unused
flaskr/marks.py:127:58: W291 trailing whitespace
```

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

Incorrect

Figure 33 - Flake8 Tests

```
PS D:\onedrive\1. University of Essex\4.0 Secure Software Development\flask\flask-tutorial> flake8 flaskr/_init__.py
PS D:\onedrive\1. University of Essex\4.0 Secure Software Development\flask\flask-tutorial> flake8 flaskr/api.py
PS D:\onedrive\1. University of Essex\4.0 Secure Software Development\flask\flask-tutorial> flake8 flaskr/db.py
PS D:\onedrive\1. University of Essex\4.0 Secure Software Development\flask\flask-tutorial> flake8 flaskr/assignments.py
PS D:\onedrive\1. University of Essex\4.0 Secure Software Development\flask\flask-tutorial> flake8 flaskr/auth.py
PS D:\onedrive\1. University of Essex\4.0 Secure Software Development\flask\flask-tutorial> flake8 flaskr/marks.py
PS D:\onedrive\1. University of Essex\4.0 Secure Software Development\flask\flask-tutorial> 
```

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 Delete (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](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

Codecademy. (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 \(www.gov.uk\)](https://www.gov.uk/government/publications/providing-remote-education-guidance-for-schools) [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 Lombardi, 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.