TOPICUS 6: *Giovanni Falsetti, Gustavo Silva Prado, Muhammad Reynard Enriza, Samer Saleh, Vladislav Mirzoev*

Assignment 3

Security analysis:

This security analysis report gives a general overview of the security measures established on our website, with an emphasis on how sensitive data like Burgerservicenummer (BSN) and passwords are handled. The paper identifies both the benefits as well as the drawbacks of the security measures implemented on our website.

BSN and passwords are important, sensitive bits of data that need to be well-protected. It's essential that our website secures sensitive data using the SHA-256 hashing technique. It is crucial to remember that SHA-256 has certain drawbacks and problems despite being commonly utilized. As technology and processing power have developed over time, SHA-256 has become more vulnerable to attacks like collision and preimage attacks. It is better to apply hashing algorithms that are stronger and more secure and that are made especially for password hashing such as Argon2.

An important security safeguard applied on our website for preventing SQL injection attacks is the usage of prepared statements. By separating the SQL code from the user input via prepared statements, malicious users find it more challenging to introduce malicious SQL queries. This precaution reduces the possibility of malicious input being used to gain unauthorized access to or manipulate your database. To maintain a strong security posture, it's crucial to make sure that prepared statements are constantly used across all database interactions.

It is a good security practice to use input validation to stop the infiltration of inadequate or malicious data. The possibility of security flaws like code injection can be considerably decreased by limiting user input to the necessary information. To avoid any potential exploitation, it is crucial to make sure that the validation criteria are thorough and cover every circumstance.

To conclude, the last security measure applied was the use of session storages. It prevents the users of the system from accessing URIs dedicated to other users of the system. While it improves security, it should be used in conjunction with solid session management, reliable authentication, and authorization procedures, as well as safeguards against session fixation (attacker setting a session identifier for a victim before the victim logs into a web application or website), to maintain the best possible security posture.

Limitation:

The website heavily relies on the sessionStorage, which stores the account ID of the user that is currently using the website, but the problem with this is that if a user is knowledgeable enough, they can access the sessionStorage through inspecting the elements of the website; if the user knows other emails within the system, they can access other parent's accounts and view all their information through the dashboard. A solution to this problem is by implementing authentication tokens as they apply one-way encryption to the data stored in the token, and is considered complex, so brute forcing attacks are computationally infeasible. In addition to that, they have a short validity period, which reduces the window of opportunity for an attacker to exploit an intercepted token.

Another effective security technique that would have been beneficial for our website is the implementation of input sanitization. Cross-site scripting (XSS) attacks and other types of code injection are made less likely by forbidding the insertion of invalid characters.

Software and user story testing

Software testing (JUnit tests):

Overview
In total, there are 10 resource classes, 7 of which were fully tested (the other three were not tested). The methods tested were the methods that use the crud operations. The implication faced due to testing was that for some tests where objects are being created and added to the database, if it is being tested again, then the tests will fail; since some methods check for the existence of the object before creating it. A limitation could be that some of the tests contain information/objects that exist in the database, so this can be an issue if these objects are deleted from the database. This was done since there were difficulties faced when trying to create objects as they might need to be added to multiple tables in the database, which can trigger errors due to the constraints in the database.

JUnit test - AccountResourceTest:

1. loginTest: To test the parent login, an account is created, so that is can be used in the login test. The test is to check if the account is a valid account within the database, so that when they try to login, they get redirected to the user dashboard page.
2. adminLoginTest: A similar test to loginTest has been carried out, but if the admin's account is valid, they are redirected to the registrations page.
3. getNameTest: This tests if the account ID, which is the email, maps to its corresponding user's name.

4. fetchAccountDetailsTest: This tests the details of a parent, the assertions make sure that when getting each individual element from the method, they match with the details that were being tested.

JUnit test - ParentResourceTest:

1. getChildrenTest: This tests if the result of getting all children of a specific parent contains the student specified within the test.
2. getGuardiansTest: This tests if the result of getting all the guardians contains a specific guardian that exists within the database.
3. getParentDetailsTest: This tests the result of getting the details from the database of the parent specified within the test. It checks if each individual element retrieved corresponds to the information specified in the test, and returns true if all the conditions in the IF statement are fulfilled.

JUnit test - SchoolAdminResourceTest:

1. getSchoolRegistrationsTest: The test done here makes sure that the registrations retrieved have the same school ID as the school admin, so that they only see registrations of their school only. It checks if there is any registration that has a different school ID than the admin that was specified in the method, and if true then the test fails.
2. getSchoolRegistrationsOrderedTest: This test specifically gets all registrations sorted by date and makes sure, as shown in the loop, that if the current registration has a date that is after the next registration, then the method fails since that means that the registrations are not sorted in order of date.
3. getParentStudentsTest: This test checks for the number of kids the parent specified within the test has, who already exists within the database. In the database, he has 8 children, so it is assumed that when the getParentStudents method is called on that parent, it will return the 8 students.
4. updateStatusTest: This test checks the registration status of the student specified within the test with id 7, who is already in the database. He has a status of ‘Under review’, so when the method is called to change his status, a check is made to make sure that the status has changed to ‘Accepted’.
5. updateEditTest:This test checks the editing status of the student specified within the test with id 7, who is already in the database. He has a status of ‘N’ (which means no), so when the method is called to change his status, a check is made to make sure that the status has changed to ‘Y’ (which means yes).
6. deleteRegistrationTest: This test check if deleting a registration of the student specified within the test with id 7, who is already in the database. When the method is called, a check is made to see in all the registrations if that student with id 7 still has a registration, if so then the test fails.

JUnit test - SchoolResourceTest:

1. getSchoolsTest: This tests if the schools specified in the test are returned when the method is called. It checks in all schools, retrieved by the getSchools method, that the three schools (which already exist in the database) exist within the result. If they are all found then the loop is terminated and the test succeeds.
2. findSchoolTest: This tests if the school being searched for using its id, returns a name. If it does not return the name “test1”, then it means that the school with id 1 does not exist.

JUnit test - SignUpResourceTest:

createAccDBTest: This method tests if the account created within the test is actually stored in the database. This is done by getting a list of all accounts, and check if an account is found with the same exact details, then this means that the account has been created successfully.

JUnit test - StudentResourceTest:

1. getStudentsTest: This tests if the getStudents method returns all students. For testing purposes, two new students are added, and the method is tested to see if it returns those two students. The assumption made here in the test is that if it returns these two students, then it will most probably return all students.
2. findStudentTest: This tests the findStudent method by first trying to extract a student from the list of all students. After that, their ID is taken and applied in the findStudent method as a parameter, and a check is made to make sure that the student found using the method has the name of the student that was extracted from the list of students.
3. editChildNameTest: This tests the editChildName method by first extracting a student’s details from the list of students, and the method is then applied to change the child’s name. A check is then made to see if the student still has the same id, but a different name. If so, then this test succeeds.
4. editBirthDateTest: This tests the editBirthDate method by first extracting a student’s details from the list of students, and the method is then applied to change the child’s birthdate. A check is then made to see if the student still has the same id, but a different birth date. If so, then this test succeeds.

JUnit test - TopicAdminResourceTest:

1. addSchoolTest: This tests if the school added exists in the list of all schools. After the method is called to add the school to the database, a list of all schools is checked through to make sure that the school is added successfully. If so the loop breaks, and the test succeeds.
2. removeSchoolTest: This tests if the school is deleted and does not exist anymore in the list of all schools. After the method is called to delete the school (using its id) from the database, a list of all schools is checked through

to make sure that the school is deleted successfully. If so, the loop breaks and the test succeeds.

3. updateSchoolTest: The test checks if some of the elements of a school, which already exists in the database, change when the method is called. The elements, tuition fee and contact number, are changed and the update is send to the database by calling the method. When all schools are retrieved, a check is made in a loop to make sure that the information of the school contains the updated information. If the information is not updated, the loop breaks and the test fails.
4. addAdminTest: This tests if the admin added actually exists by checking the list of all admins. The list of all admins is iterated through and if the admin is found, the loop breaks, and the test succeeds.
5. removeAdminTest: This tests if the admin deleted does not exist anymore by checking the list of all admins. The list of all admins is iterated through and if the admin is found, the loop breaks, and the test fails.
6. getAdminTest: This tests the result of the getAdmin method with the details specified in the test. If they match, then the method returns the admin's details successfully.
7. updateSchoolAdminTest: This tests if the updated information of the admin is in the database. An admin, from the database, has three updated elements: address, phone number, and password. The method is then triggered,and following that, the list of all admins is iterated through to make sure the list contains the updated details. If the admin is found, but the three elements are not updated, the test fails.

User story testing (System tests)

Story	Description	Expected result	Actual result	Pass/ Fail	Additional info
Logging in as a parent	1. Click on log in 2. Select guardian 3. Enter email 4. Enter password	Parent can now view their dashboard	Image 1 in appendix	Pass	If the parent did not have an account, then this test will fail.
Logging in as a school	1. Click on log in	Admin can now view their	Image 2 in appendix	Pass	The admin cannot create an account,

admin	2. Select admin 3. Enter email 4. Enter password	dashboard			they get their credentials from the Topicus admin.
Registering a child without having an account (parent)	1. Click on link to form. 2. Fill in all fields. 3. Submit when all fields are filled.	User gets a pop up saying registration successful.	Image 3 in appendix	Pass	
Registering a child using an account (parent)	1. Click on log in 2. Select guardian 3. Enter email 4. Enter password 5. Click on new form 6. Fill in fields 7. Submit when all fields are filled in. 8. Page will redirect to the school specific form. 9. Fill in the additional fields in the school specific form. 10. Submit when all fields are filled in.	Parent will be redirected to the child page and will find a new card with the newly registered child.	Image 4 in appendix	Pass	
Downloading a registration (school admin)	1. Click on log in 2. Select admin 3. Enter	Admin will see the registration being downloaded as a pdf.	Image 5 in appendix	Pass	

	email 4. Enter password 5. Admin will automatically redirect to the registrations page 6. Click on a registration 7. A pop up will be displayed 8. Click on the here button.				
Accessing messages (both parent and school admin)	1. Click on log in. 2. Make sure that you click on your role in the website 3. Fill in credentials 4. If you logged in as a parent, Click on messages. If you logged in as an Admin, click on contact parents.	Admin can see a button saying add new message and can also see other chats if there were any. Parent can see messages received from the admin regarding the registration.	Image 6 and 7 in appendix	Pass	A parent can only send a message if an admin sent them a message regarding the registration of their child.
Accepting a registration (school admin)	1. Click on log in 2. Select admin 3. Enter email 4. Enter password 5. Click on a		Image 8 and 9 in appendix	Fail	Admin must fill in a message and a subject to be able to accept a student's registration.

	registration 6. A pop up will show up. 7. Click on send a message to parent 8. Click on drop down list 9. Select accept				

Appendix

Image 1

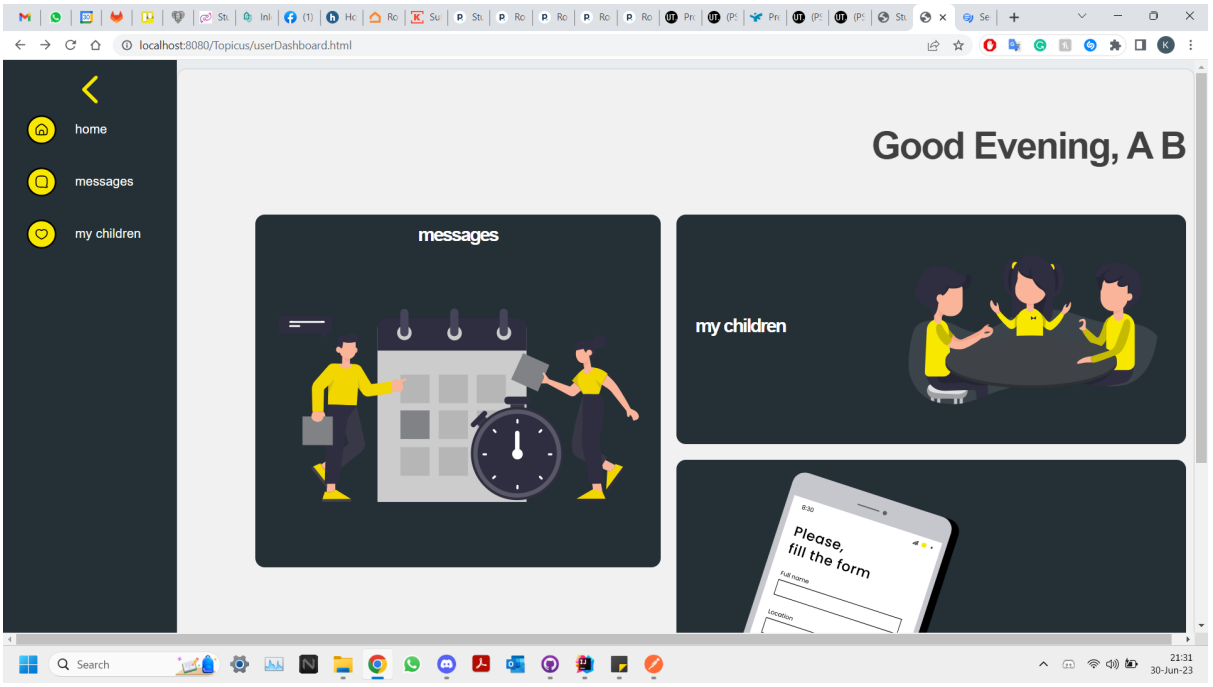


Image 2

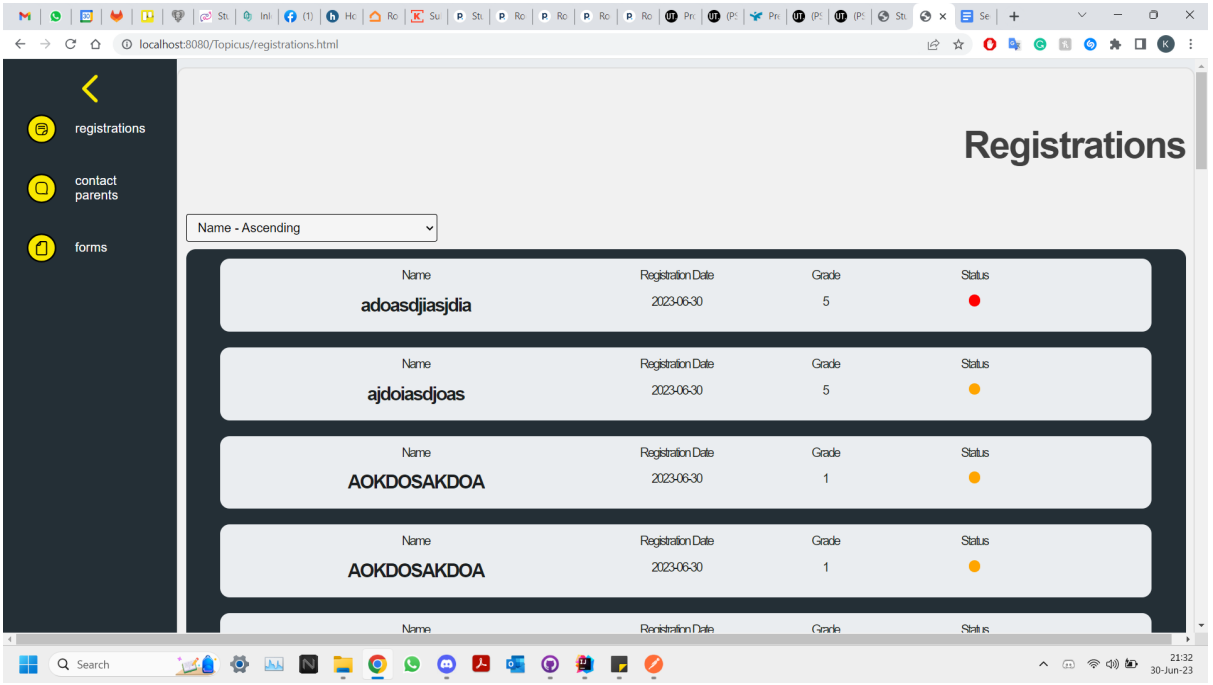


Image 3

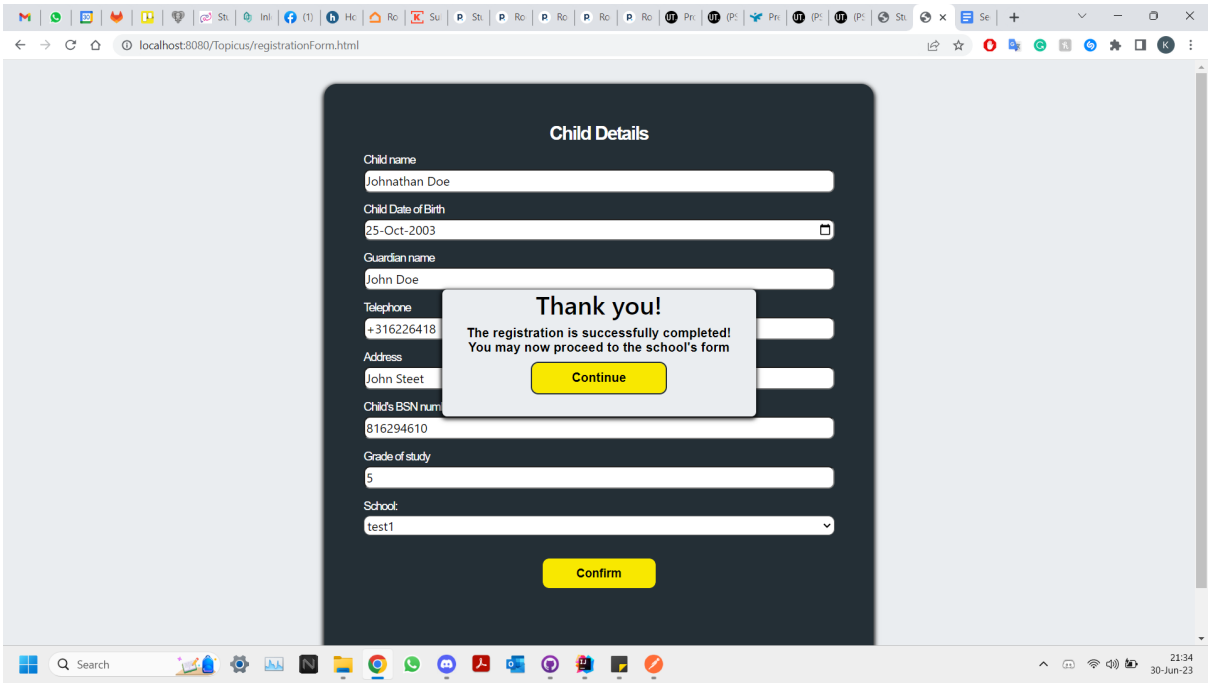


Image 4

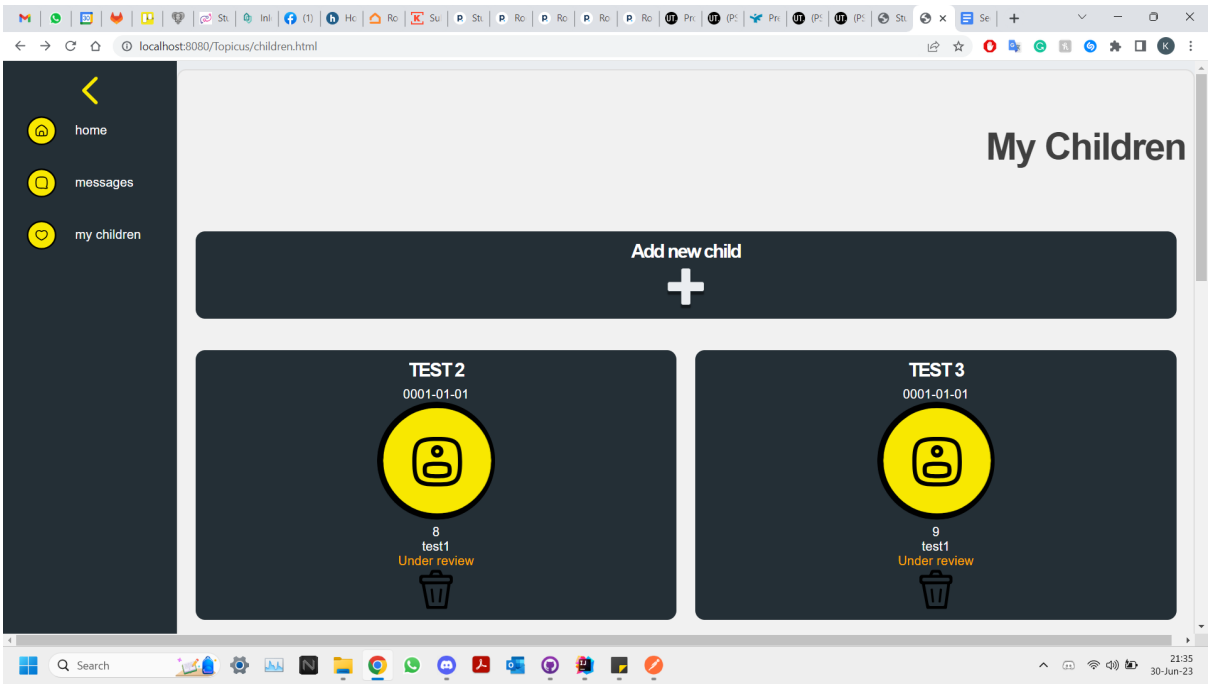


Image 5

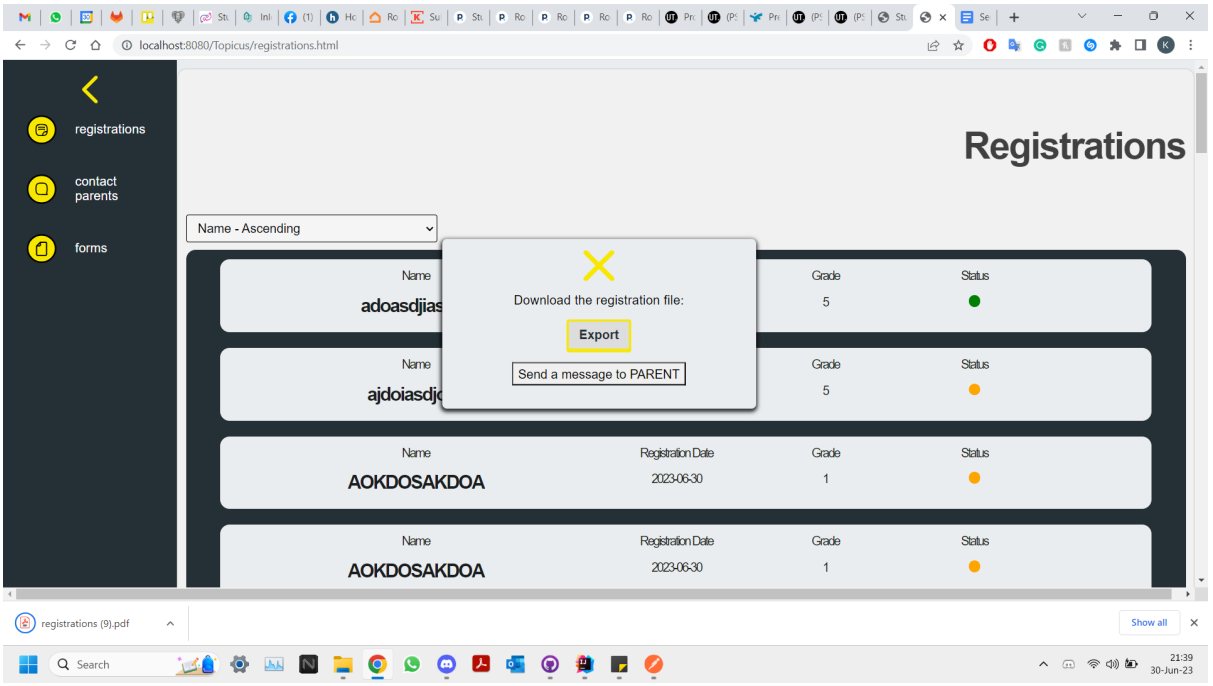


Image 6 (parent messages)

TOPICUS 6: *Giovanni Falsetti, Gustavo Silva Prado, Muhammad Reynard Enriza, Samer Saleh, Vladislav Mirzoev*

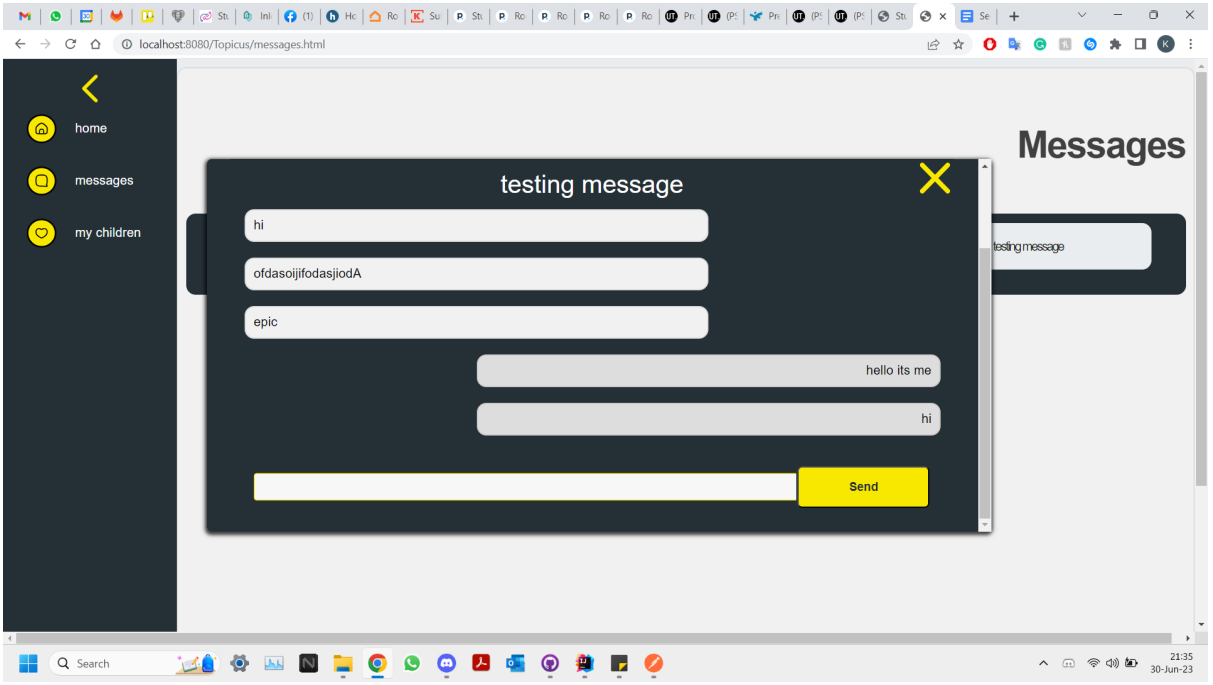
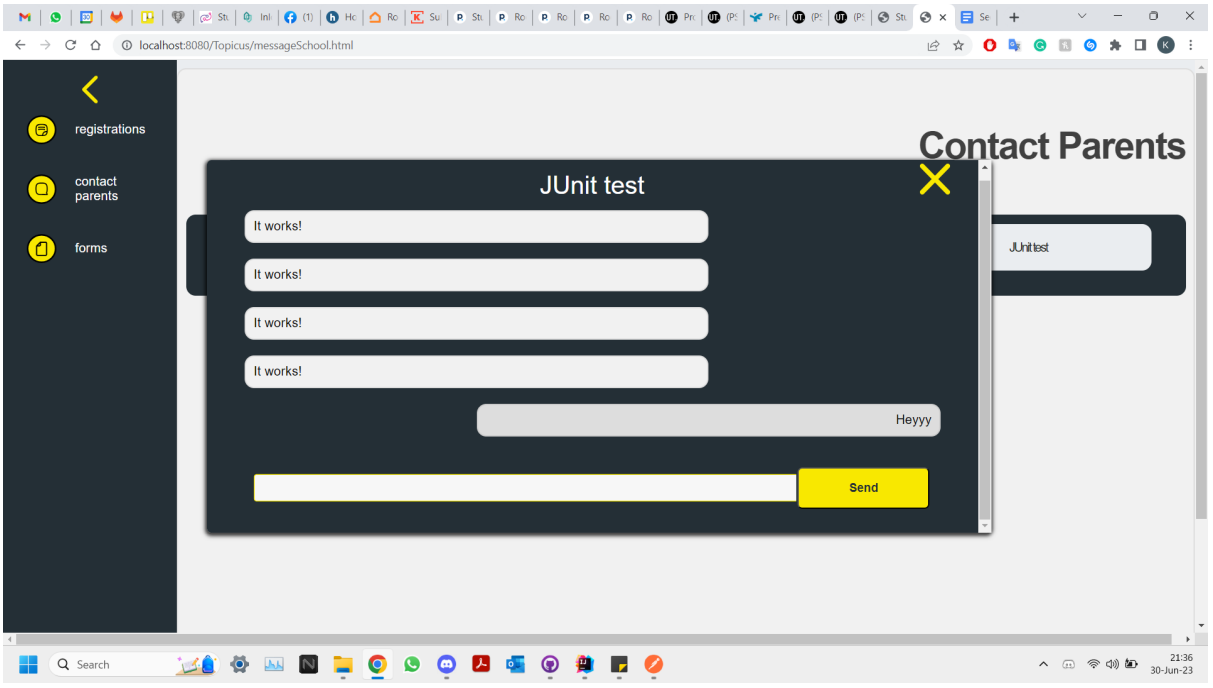


Image 7 (admin messages)



TOPICUS 6: *Giovanni Falsetti, Gustavo Silva Prado, Muhammad Reynard Enriza, Samer Saleh, Vladislav Mirzoev*

Image 8

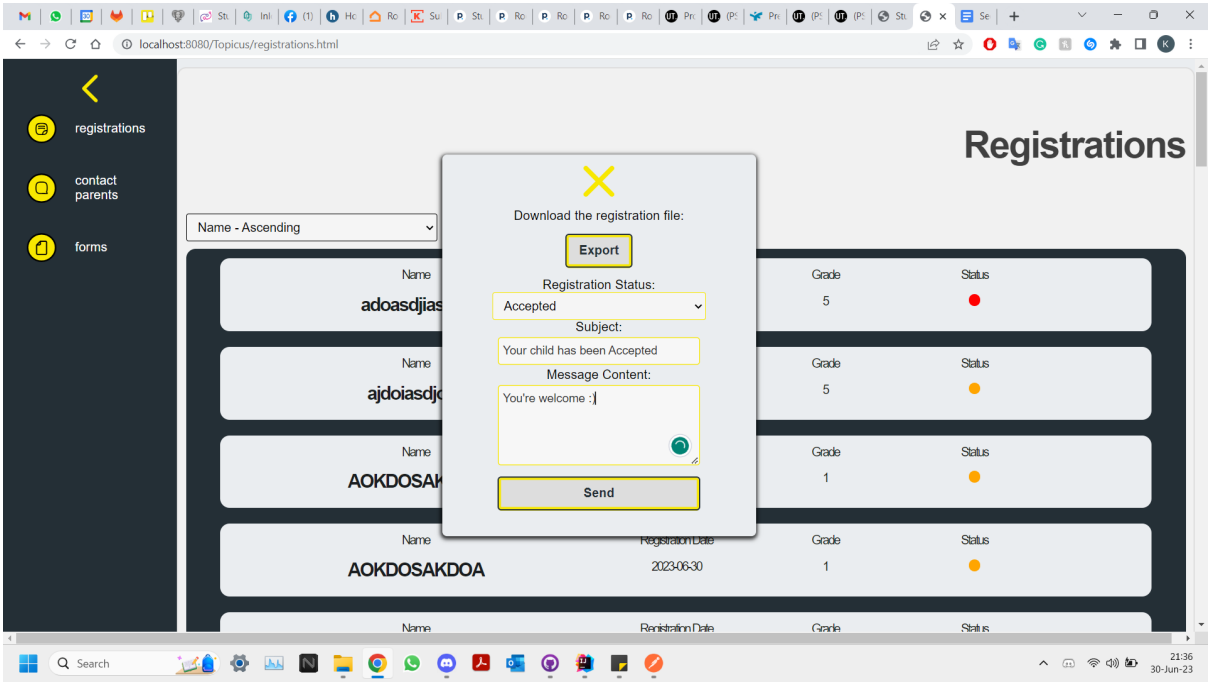
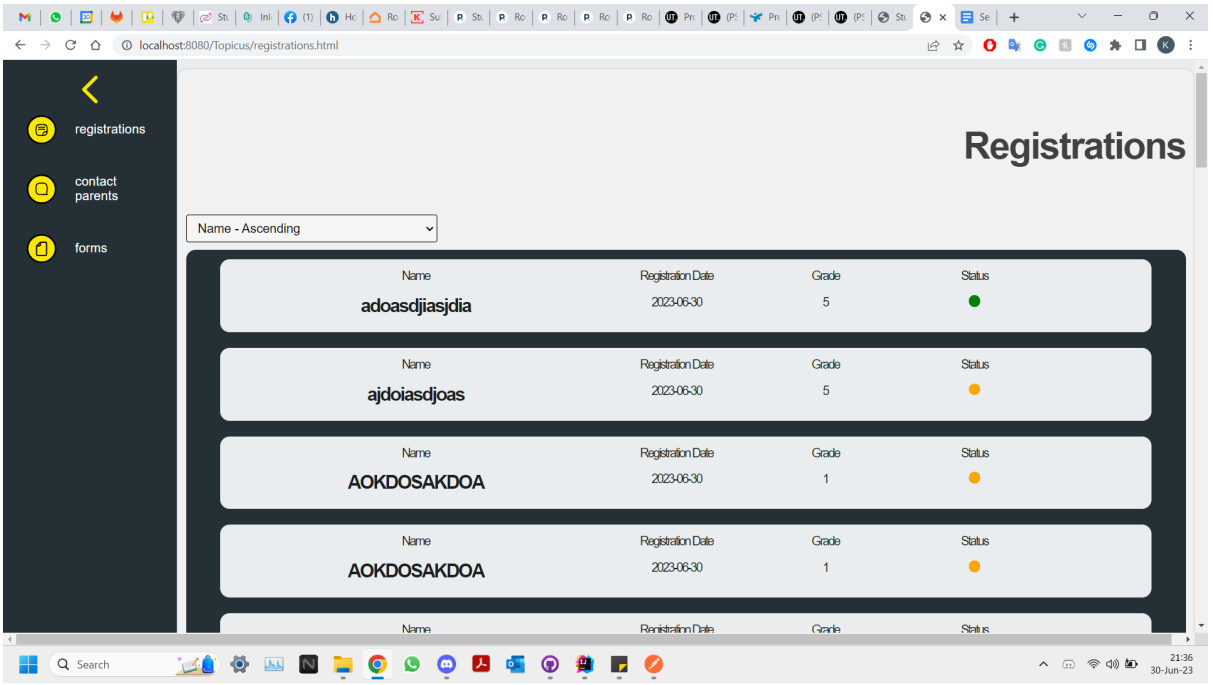


Image 9


localhost:8080/Topicus/registrations.html
registrations
contact parents
forms
Registrations
Name - Ascending
Name
adoasdjiasjdia
Registration Date
2023-06-30
Grade
5
Status
Name
ajdoiasdjoas
Registration Date
2023-06-30
Grade
5
Status
Name
AOKDOSAKDOA
Registration Date
2023-06-30
Grade
1
Status
Name
AOKDOSAKDOA
Registration Date
2023-06-30
Grade
1
Status
Search
21:36
30-Jun-23