THE UNIVERSITY OF SYDNEY

ELEC3609/ELEC9609 Group Assignment

Assignment 4: Deployment, Security and Testing (15%)

For Assignment 4, you are required to complete your project development, including security implementation and testing. Additionally, you must deploy all components of your Django project to a production environment on an AWS EC2 instance. The deadline for this assignment is 1st November 13:00 (Friday, Week 13).

Assignment Submission

Code Submission

For the final assignment, all updates related to your project, including security measures, testing, or any additional feature modifications, **must be pushed to Uni GitHub** before the deadline. This includes any changes made after Assignment 3 as part of this final phase.

Please note that while you are allowed to make changes to the code for **security and deployment purposes**, the overall project mark (35 in total) will not change based on these modifications. The changes you make will contribute **only to Assignment 4** and will not affect the marks you received for previous assignments.

You do not need to submit the code on Canvas. Your last commit on the main branch before the deadline will be considered your formal submission for this assignment.

Deployment Specifications

Your group will deploy your application to the provided Amazon Web Services (AWS) account. You must create a Linux-based instance on AWS to host your website. Do not expect a graphical environment, you will need to configure this server from the command line. In this unit of study, **you are provided with an AWS education account with $100 credit**. A complete step-by-step guide will be available on Canvas later.

For each group, **it is only required to deploy the application on one member's AWS Education account.** There is no need to deploy the application on all group members' AWS accounts. Make sure that the deployment is fully functional on the chosen account, as this will be the version assessed during the in-session marking.

Your team is required to deploy the web application on a server instance with specifications **similar to** the following (which are roughly equivalent to an AWS t2.micro instance):

- **Operating System**: Ubuntu
- **CPU**: 1 Virtual CPU (Intel Xeon)
- **RAM**: 1 GiB RAM
- **Storage**: 8 GiB Storage (EBS)

While these are the recommended specifications, you are free to use other setups as long as your application performs correctly and efficiently within the chosen environment.

In-Session Marking

As part of Assignment 4, deployment and part of security evaluation will be assessed live during your Week 13 lab session. **No documentation or report submission is required for deployment and security.** Each group is expected to be given 10 minutes for showcasing your deployment. The Q&A session following your demonstration will cover deployment and security aspects, with questions randomly cutting across all sections.

An individual will receive 0 marks if they are absent for the demonstration. Additionally, a penalty will be applied to the group if the team fails to adequately answer questions during the demonstration.

Testing Report

You are required to submit a testing report to Canvas. This report should cover the key aspects of your testing strategy, including unit and integration tests. For detailed information on what to include, please refer to the following sections of the assignment.

Late Submission

Late submissions will be penalised 20% per day late, including weekends. Plagiarism will not be tolerated, and your assignment may be submitted to a plagiarism checking service.

Note

While there is no formal submission for deployment documentation, **it is highly recommended that you write down notes** and prepare for any deployment or security-related questions in advance.

The cost of using cloud services can be tricky to figure out. If you're not careful and don't know a lot about how they work, you might end up paying extra by mistake. You can check out AWS pricing calculator before setting up your production environment. **If you use any outside cloud services, it is your responsibility to make sure you never put in your credit card details on any of them.**

Part 1 Deployment (5 marks)

By the end of Week 13, your application will be running in a production environment in the cloud. Deploying to production always takes longer than expected. Do not leave deployment to Week 13, you should have your production environment running (in a basic capacity) several weeks prior.

Deployment Requirements

For your deployment, you are required to complete both **First-Tier** and **Second-Tier** tasks to ensure the successful setup and security of your application. Each task plays a critical role in verifying the stability and functionality of your deployment. All tasks listed below can be examined during the in-session marking.

First-Tier Tasks	Second-Tier Tasks
Deploy the project on an AWS EC2 Linux instance.	No errors are identified in the deployed application.
Allow traffic to your instances on ports 22 (SSH), 80 (HTTP), and 443 (HTTPS). Must configure Nginx as the reverse proxy.	Configure the server to automatically serve requests after a reboot, without manual intervention.
Use uWSGI to run the Django application.	Disable Django's debug mode in the production environment.
Ensure all static files (i.e., JS, CSS, images, media) are correctly loaded in the application.	Optimise server load by configuring Nginx to serve static files (i.e., JS, CSS, images, media).
Verify that most of the application's functionalities work correctly in production environment.	Ensure your infrastructure (server host, database, and other resources) is configured with basic security measures such as SSH access control, appropriate firewall rules, and limiting access to necessary ports.

Marking and Submission

You are not required to submit anything for the actual deployment; instead, your tutor will assess all above tasks during the lab in Week 13. You must ensure your application server has a **functioning terminal-based frontend to your database** (e.g., psql or mysql) to allow markers to review your database state and structure.

Please be aware that your team will have exactly only one opportunity to present your deployment work to the tutor for evaluation during the week 13 lab. Make sure you are prepared before the in-session marking.

Part 2 Security (5 marks)

As part of Assignment 4, you are required to implement the security measures discussed in the lectures. Your development must be protected against common vulnerabilities and follow best practices for securing web applications. This includes ensuring proper access controls, configuring firewall rules, and safeguarding your infrastructure.

Security Requirements

You are expected to implement and meet the following security perspectives to ensure your application is robust and secure:

1. Your load balancer and Django application must handle **5 concurrent users** (5 users viewing and using your website simultaneously).
2. Your **admin panel** must not have users with simple passwords (“password1”, “test”, “admin”, etc.)
3. The **database user(s)** used by your application must not be the root or system DB user, and should not have permissions to CREATE, ALTER, or DROP tables. In case of SQL injection, the permissions should be limited. You may also consider removing DELETE/UPDATE privileges if appropriate. If using SQLite, ensure the database file is protected by proper permissions.
4. Use **HTTPS/SSL** to prevent transmission interception and modification. Services like ZeroSSL and Let’s Encrypt offer free SSL certificates.
5. **Django’s debug mode must be disabled** (DEBUG=False) in the production environment.
6. **Application-level errors/bugs** should not expose a stack trace.
7. Ensure there are **no serious application-level bugs** and no issues with user access permissions.
8. **User authentication** must be secure. Your implementation of Django’s built-in user authentication will be checked for vulnerabilities, and we may attempt to modify cookies or JWTs to test for weaknesses.
9. Your application must be protected against common vulnerabilities such as **XSS, SQLi, and others** discussed in Lecture 7.
10. It should not be possible for an external user to access your **Python source code** or load sensitive files (e.g., /etc/password) through a browser or HTTP request.
11. Ensure passwords cannot be discovered by looking at the server’s **command history**.
12. The **private key** file downloaded from AWS for logging in with the student user should not be present on the server.
13. Use a **private key file** for SSH authentication, rather than password-based logins.

Marking and Submission

During the in-session marking, tutors will perform some random check on your project. The security performance of your project will also **be evaluated again** after the in-session marking.

You will need to **commit any changes related to security** to your GitHub main branch before the deadline. The last commit before the deadline in the main branch will be treated as the formal submission.

Part 3 Testing (5 marks)

You are required to test both the frontend and backend of your project. For the backend, you **must write unit tests** to verify the functionality of your Django application. This includes testing models, views, forms, and templates to ensure proper data validation, response status codes, and core functionalities (Actually, all aspects).

For the frontend, **unit tests are not required**, but you must demonstrate the types of work you've completed to reduce errors and ensure the user interface functions correctly, such as manual testing for form submissions, validation, and key functionalities.

Testing Report Requirements

You are required to submit a comprehensive testing report for **both the frontend and backend** of your project. Your report should at least include the following elements:

1. **Test Case Design**: Explain why you designed test cases and what specific functionality those intended to verify.
2. **How the Tests Work**: Describe how the tests operate, including how they simulate real-world use cases and cover critical parts of your application.
3. **Coverage**: Outline which aspects of the project were covered by your tests, including backend unit tests and any manual checks performed for the frontend.
4. **Results**: Summarise the outcomes of your tests, indicating which tests passed or failed, and highlighting any issues or bugs that were discovered and fixed during the testing process. Include a coverage report for the backend as well.

Marking and Submission

For marking and submission, you are required to submit two things:

1. **Python Unit Test Cases**: All Python unit test cases for your backend must be pushed to your GitHub repository before the deadline. Ensure that the tests are well-organised, properly commented, and included in the main branch.
2. **Testing Report**: The testing report should be uploaded to Canvas.

Marking Criteria

	Novice	Competent	Proficient
Deployment	0: Any of the first-tier tasks are incomplete or incorrect.	2.5: All first-tier tasks are successful but any of the second-tier tasks fail.	5: All tasks from both tiers are successfully completed.
Security	0-3: The project fails two or more security requirements.	3-5: The project fails only one security requirements.	5: The project passes all security requirements without any failures.
Testing	0-1: Minimal or no testing has been conducted. Little to no effort is shown in both backend and frontend testing.	1-3: Sufficient testing has been done with some critical functionalities covered. A few edge cases are addressed, and the test coverage is moderate. The unit test code is functional but lacks clear organisation. A reasonable amount of effort is demonstrated for frontend testing.	3-5: Comprehensive testing has been conducted, covering all key functionalities, with most edge cases considered. Test coverage is high, and the unit test code is both well-organised and thoroughly commented. Significant effort is demonstrated for frontend testing, including browser testing, manual checks, and debugging. The report includes clear and detailed documentation of testing work.