Task 9.1 Technology Stack - Cloud Group 5

Brief Opening:

The project will be designed to be created in two parts, A and B. Within the single project, the introduction of programmable language, database design, and backend development for the team will take precedence over other departments within part B of the design. Part B of the project will include all other facets of the final product, for example caching, frontend, web server, monitoring, and security. The main functions of the website will be retrieving data from the users and data being created from the owners of the website, and this forms the basis of the development of the website.

Group 5 has also created an alternative technology solution to the first design, one that focuses on prioritising security and possibly more familiar technology that could be seen as an easier solution to long term scalability. Both technologies implemented in Technology Stack A and B solve the request of the client of the project whilst also giving options for how they prefer different technologies that are integrated within their overall idea.

Technology Stack A:

- Programming language: Ruby (Ruby on Rails Framework)
- Database: MySQL (Relational Database)
- Frontend: HTML5, CSS3, JavaScript
- Caching: Memcached
- Web Server: Apache HTTP Server
- Operating System: Linux
- Security: MD5+Token
- Monitoring: Grafana
- Cloud Provider: AWS EC2 server (General Purpose)

Technology Stack B:

- Programming language: JavaScript, Python
- Database: MongoDB (Non-Relational Database)
- Frontend: HTML5, CSS3 (Bootstrap framework), JavaScript (Angular framework)
- Caching: Redis
- Web Server: Nginx
- Operating System: Windows
- Security: MD5+Token, Yubikeys
- Monitoring: Sentry
- Cloud Provider: AWS EC2 server (General Purpose)

Justifications for selecting certain technologies in Stack A vs B

	Stack A	Stack B
Cost	**Ruby/Ruby on Rails** - $0 **MySQL** - $0 **Memcached** - $0 **Linux** - $0 **Grafana** - $0 **Apache** - $0 **MD5+Token** - $0 **AWS** - $0.0333 (hourly)	**JavaScript** - $0 **Python** - $0 **Windows 10 Pro** - $339 (One Time) **Sentry (Team)** - $26 (monthly) **Redis**- $0 **Nginx** - $2500 (yearly) **MongoDB (40GB)** - $0.75 (hourly) **MD5+Token** - $0 **Yubikeys** - $45 *no. of employees **Sentry** - $80 (monthly) **AWS** - $0.0448 (hourly)
Skill level required	**Ruby** - extremely beginner friendly, natural to read, and easy to write, which means faster productivity. RubyGems can help with implementing authentication, detecting vulnerabilities, and rooting out bugs etc. **MySQL** - some team members have knowledge of MySQL and it is relatively simple to learn and implement. **HTML/CSS/JavaScript** - relatively easy to learn and improve skills. **Memcached** - designed to be simple and generic and therefore the team would find it easy to implement. **Apache HTTP Server** - Apache is well-known and easy to configure, and doing some research before implementation would allow for the correct implementation. **Linux:** open-source OS, easy to set up and modify if required. **MD5+ Token:** Easy to implement as MySQL supports MD5 password hashing. **Grafana:** simple open-sourced analytics and monitoring for web applications. Quick to deploy. **AWS:** AWS makes deploying web	**JavaScript/Python** - JavaScript and Python are a few of the easier programming languages to read and understand, and would work well for our team if we had to use this one of these. Both JavaScript and Python have libraries and frameworks that can help with implementation. **MongoDB** - without some knowledge of databases, MongoDB becomes more difficult to implement, so with some research it should be easier to implement if we used it. **HTML/CSS/JavaScript** - relatively easy to learn and improve skills. **Redis** - it is made up of common data structures that most developers are familiar with. **Nginx** - determining the configurations for setting up the web server would be the hardest part, especially if the team does not have this knowledge. **Yubikeys** - easy for employees (and if wanted, customers) to obtain and use. **Sentry:** makes logging and error handling simple**.** **AWS:** AWS makes deploying web applications to the cloud simple, with websites coming online almost instantly after launch. **Windows:** one of the most common OS,

	applications to the cloud simple, with websites coming online almost instantly after launch.	easy to set up as a web server. **MD5+Token:** Easy to implement as MongoDB already supports MD5 password hashing.
Scalability	**Ruby** - demonstrates opportunity for horizontal scaling. **MySQL** - usually replaced by PostgreSQL or MongoDB to accommodate large quantities of data when scaled (but can scale up). **HTML/CSS** - scalability includes responsive design which needs to be considered when creating a website and this can be achieved with HTML/CSS. **Memcached** - the distributed and multithreaded architecture means it is easy to scale and can be scaled up and scaled out. **Apache HTTP Server** - can be easily scaled using AWS **AWS** - has an auto-scaling feature to ensure the correct number of EC2 instances are being used and can launch or terminate if and when they are needed or not needed.	**JavaScript/Python** - JavaScript is flexible and scalable, but Python is less scalable in terms of performance and execution speed. Python does not support multithreading which impacts on the performance. **MongoDB** - non-relational DB that is better for scaling out. **HTML/CSS** - same as Stack A **Redis** - it stores data in the server's main memory which ensures high availability and scalability of services and application workloads (implementation of Redis Cluster could also help with scalability). **Nginx** - this web server is able to scale to hundreds of thousands of concurrent connections on modern hardware. **AWS** - same as Stack A.

Overall, one of the main reasons for the technologies chosen in Stack A is the low cost they offer, whilst still providing good services and features. Most of the technologies within both Stack A and B are relatively easy to implement, but for Stack B the extra security features may require external advice or support for implementation. The technologies will enable us to develop the website with all the requirements set by the client, with the ability for further expansion. The selection of MySQL in Stack A offers the ability for each of the tables storing various types of information to only be accessed by those who need to. If necessary, we could incorporate technologies and features from both Stack A and B.

SWOT Analysis of both Stack A and B

STACK A

STRENGTHS	WEAKNESSES
• Low cost for the technologies • Technologies are relatively easy to implement • Relational databases can allow different security for each table (user permissions)	• Limited knowledge of creating a web application • Scalability more limited than Stack B • Limitations could be present in the structure of relational databases
OPPORTUNITIES	**THREATS**
• Use the information and learned skills to create a mobile application • Implement more security measures (potentially those mentioned for Stack B) • Could create additional features such as a chat function	• SQL injection • Cross-site scripting attacks • DDoS attacks • Information leakage

STACK B

STRENGTHS	WEAKNESSES
• More security features • Non-relational database can handle structured and unstructured data • Can be easily scaled out and up and can also handle different types of data - making it useful if the website features change in the future	• Higher cost to implement and continue operating • Extra app to carry and pay for • May need to seek advice on implementation which will cost more money
OPPORTUNITIES	**THREATS**
• Data storage capacity • More possibilities for scaling (can accommodate for more users and potentially more functions of the website)	• Loss of device (Yubikey) could lead to a breach of the system • Remote command/code execution • Database must be properly configured to prevent information leakage and unauthenticated access

The above tables show that Stack A and B have some similar strengths, weaknesses, opportunities, and threats. The limited scalability of the database in Stack A could be replaced by Stack B's database if the client is expanding the data stored on the website or the features of the website. For a few of the threats faced by using either technology stack, these could be mitigated by implementing additional security features and ensuring staff who maintain the website are aware of the threats.