## **Resourcing Estimation Summary**

| Resource | Purpose | Why it's needed |  | Specification |  | Unit Price | Estimated Quantity | Approx Cost/month |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| EC2 Instance | Server to host the application | The application needs a server (the computing power and resources) to run on |  | t3.2xlarge |  | $0.03895/hr | 1 | $280 - $300 |
| RDS Instance | Database to store user data | The application needs a database to store user data |  | db.t3 xlarge (Single-AZ) |  | $0.312/hr | 1 | $220 - $280 |
| EBS Volume | Storage for the database | The application needs storage to store the database |  | 100 GB General Purpose SSD |  | $0.1/GB-month | 100 | $10 - $20 |
| Storage (S3) | Stores files and data | Provides a secure platform to store files and data used by the application. |  |  |  | $0.023 per GB-month and $0.10 per million I/O requests |  | $23 + $0.1 (per million I/O requests) |
| Load Balancer | Distributes incoming traffic | To ensure the application can handle high levels of traffic and prevents any one server from becoming overwhelmed. |  | Elastic Load Balancer |  | $0.0225 | 1 | $16 - $50 |
| Auto-scaling | Automatically scales resources | To ensure the application has the resources it needs to handle changing levels of traffic and usage. |  | Auto Scaling Group |  | Included in EC2 cost |  | Included in EC2 cost |
| Elastic IP | Static IP for the EC2 instance | The application needs a static IP for easy access and to avoid IP changes |  | 1 Elastic IP |  | $0.01/hr | 1 | $7.30 - $10 |
| Networking | Connects the application to the internet | Enables users to access the application from anywhere with an internet connection. |  |  |  | Included in ELB cost |  | Included in ELB cost |
| Firewall | Virtual private network | To create a virtual network and limit access to the corporate network or firewall |  | 1 AWS Virtual Private Cloud (VPC) |  | $0.01/hr | 1 | $7.30 - $20 |
| Security Group | Network security | To limit access to the application to only users within the corporate network or firewall |  | 1 Security Group |  | $0.01/hr | 1 | $7.30 - $20 |
| Data Transfer | Transfers data in and out of the app | Enables data to be transferred in and out of the application as needed, such as when users upload or download files. |  | Inbound & Outbound Data Transfer |  | $0.09 per GB - $0.01 per GB |  | $0.09 to $0.01 per GB |
| Backup | Protects data from loss | Ensures that the application's data is protected in case of unexpected disruptions or failures. |  | Backup and DR solutions |  | $0.09 per GB-month |  | $9 |
| DevOps Engineer | Server maintenance and management | To ensure the server runs smoothly and is updated as required |  | 1 DevOps Engineer |  | Included in WPM budget | 1 | Included in WPM budget |

Note that above table is a summary of key resources and the related cost, manpower and AWS services required to host the application on AWS, based on the assumption that the application will serve about 100 users and data storage requirements will not exceed 100GB. The cost and manpower required will vary depending on the specific requirements and usage of the application.

It is recommended to use the AWS pricing calculator to get a more accurate and up-to-date estimate of the cost of the resources and services required. Also, it is important to note that we need one DevOps Engineer to handle the management and scaling of these resources (which is assumed is already costed into WMP manpower resourcing).

## **Frontend Interface Specifications**

| Technical Requirements | Description |
| --- | --- |
| Drop-down menu | A drop-down menu should be provided on the frontend interface, which allows the user to select specific names from a predefined list. |
| Run button | A button should be provided on the frontend interface, which allows the user to initiate the backend logic and process the data based on the selected name. |
| Progress indicator | A visual indication should be provided on the frontend interface, which shows the progress of the backend logic and displays a "running in progress" message to the user. |
| Download button | A button should be provided on the frontend interface, which allows the user to download the generated report in Excel format. |
| Data Refresh | The backend should pull data from Hadoop cluster monthly to ensure that most up-to-date information is used for report generation. Add a cronjob to refresh the data monthly. |
| Authentication & authorization | Only authorized users should be able to access the frontend interface and initiate the backend logic. |
| Excel report generation | The backend logic should generate an Excel report based on the selected name and the data from the Hadoop cluster. |
| Caching | To improve performance and reduce the load on the backend, caching techniques will be used to store frequently requested data on the frontend and avoid unnecessary requests to the backend. |
| Error handling | The backend will have appropriate error handling mechanisms in place to handle any errors that may occur and provide meaningful responses to the frontend. |
| Monitoring & Logging | Monitoring and logging should be in implemented to track the performance of the backend and troubleshoot any issues that may arise. |
| Scalability | The backend should be designed to scale horizontally and vertically as the number of users and amount of data increases. |
| Hosting infrastructure | The web interface will be hosted/URL on a dedicated hostname within the corporate firewall. |
| Firewall configuration | The corporate firewall should be configured and whitelisted to allow access to the dedicated hostname. Configure the network to allow access to the application hostname within the corporate firewall |
| Domain Name System (DNS) | Create DNS records to map the dedicated hostname to the IP address of the application |
| Network security | The application will leverage existing corporate network security measures to protect the application against unauthorized third-party access |

## **Other technical considerations:**

* **API design:** The backend should be designed to expose a well-defined and easy-to-use API that the frontend can use to access and manipulate the data. This may include designing endpoints for specific data operations and using appropriate HTTP methods and status codes. The backend and frontend should agree on a common data exchange format, such as JSON, to facilitate seamless data exchange between the two domains.
* **Security:** The backend should implement appropriate security measures to protect sensitive data and prevent unauthorized access. This may include implementing authentication and authorization mechanisms, such as tokens, as well as encrypting data in transit and at rest.