Prompts:

Proposal Document Prompts:

1. Act as a Cloud Architect and Project Manager. Help me define the project "CampusBites," a canteen pre-ordering system that uses a 3-tier serverless container architecture on AWS to solve manual queuing issues.
2. Write an Executive Summary for the CampusBites proposal, highlighting the use of Amazon ECS, AWS Fargate, and Amazon ECR to modernize university cafeteria operations.
3. Formulate a Problem Statement for a manual university canteen system, focusing on peak-time congestion, staff inefficiency, and the revenue loss associated with unverified "prank" orders.
4. Define the Software Requirements for the CampusBites app, splitting them into Functional Requirements (Auth, Menu, Kitchen Dashboard) and Non-Functional Requirements (High Availability, Scalability).
5. Specify how Student/Staff ID authentication and real-time order status tracking will be implemented as functional requirements to ensure accountability and efficiency.
6. Detail the AWS Cloud Architecture by listing essential services such as ECR for image storage, ECS for orchestration, and AWS Fargate for serverless compute.
7. Integrate a PostgreSQL RDS instance and AWS Secrets Manager into the service breakdown to handle data persistence and sensitive credential management.
8. Explain the roles of the Internet Gateway and NAT Gateway in facilitating secure external and internal networking for containerized tasks.
9. Design a deployment strategy called "Defense in Depth," isolating the system into public and private subnets across multiple tiers.
10. Elaborate on the Web Tier (Frontend) setup using React.js as containerized Fargate tasks in a Public Subnet accessible via an Application Load Balancer.
11. Describe the App Tier (Backend) setup using Node.js in a Private Subnet, ensuring it is isolated from the public internet and only accepts traffic from the ALB Security Group.

12. Detail the configuration of the Data Tier, focusing on a managed PostgreSQL RDS instance in an isolated private subnet with restricted inbound access.
13. Create a section on Security and Secrets Management that explains how ECS tasks use IAM Roles to pull DB passwords from Secrets Manager at runtime.
14. Add a section for Administrative Access featuring a Bastion Host (EC2) in the Public Subnet to serve as a secure bridge for database maintenance.
15. Specify the access control rules for the Bastion Host, including whitelisted administrative IP addresses and SSH tunneling for PostgreSQL management.
16. Draft a Conclusion for the project proposal that summarizes the benefits of transitioning from EC2 to a modern, scalable ECS Fargate architecture.
17. Review the entire proposal to ensure all three group members (Muhammad Samer Nisar, Ahmad Umar Khan, and Mirza Muhammad Rehan) are correctly listed on the cover page.
18. Generate a finalized version of the Project Proposal titled "CAMPUSBITES: A High-Availability Canteen Pre-order System Deployed on AWS," ensuring all technical components are properly integrated.

AWS Architecture Implementation Prompts:

1. Provide a detailed, step-by-step guide on how to create a custom Virtual Private Cloud (VPC) for a 3-tier architecture. Use the CIDR block 10.0.0.0/16 and ensure the configuration is optimized for the us-east-1 region.
2. Walk me through the process of creating 6 specific subnets within my VPC. I need 3 subnets in us-east-1a and 3 in us-east-1b to cover my Public, Private App, and Private Data tiers. Please include appropriate CIDR assignments for each.
3. Provide the exact AWS CLI commands required to create an Internet Gateway (IGW) for my VPC and then attach it to my specific VPC ID to enable external connectivity.
4. Explain the practical steps to provision an Elastic IP and create a NAT Gateway within one of my public subnets. This is necessary to provide controlled internet access for the tasks running in my private subnets.
5. Explain how to configure a Public Route Table so that it correctly routes all outbound traffic (0.0.0.0/0) through my newly created Internet Gateway.

6. Provide a detailed explanation of how to configure Private Route Tables for my App Subnets. The configuration must ensure that all non-local traffic is routed through the NAT Gateway.
7. List the specific steps in the AWS Console to create an RDS DB Subnet Group. This group must include the two private data subnets spanning two different availability zones.
8. Provide a detailed walkthrough and the specific AWS CLI commands to create an External Security Group for my project. Name it 'External-ALB-SG' and ensure it allows inbound traffic on port 80 (HTTP) and port 443 (HTTPS) from any source (0.0.0.0/0).
9. Explain the process of creating a Security Group for my Web Tier. This group should be configured to accept inbound traffic only on port 80, and the source must be strictly limited to the Security Group ID of my External Application Load Balancer.
10. Provide a technical guide on setting up a Security Group for an Internal Application Load Balancer. It needs to accept traffic from the Web Tier Security Group to facilitate secure cross-tier communication.
11. Detail the requirements for an App Tier Security Group. Configure it so that it only accepts inbound requests on port 8080, with the source restricted to the logical ID of the Internal ALB Security Group.
12. Explain how to configure a Security Group for an Amazon RDS PostgreSQL instance. It must allow inbound traffic on port 5432 from two specific sources: the App Tier Security Group and the Bastion Host Security Group.
13. Provide the specific steps and rules needed to create a Bastion Host Security Group. The inbound rules must be locked down to allow SSH (port 22) traffic only from my specific personal IP address.
14. Guide me through the process of launching a t3.micro EC2 instance in a public subnet to serve as a Bastion Host. Include best practices for key pair management and initial security hardening.
15. Provide a comprehensive guide to provisioning an Amazon RDS PostgreSQL instance using the db.t3.micro class. Ensure it is placed in the private data subnet group and has Multi-AZ deployment disabled for this phase.

16. Walk me through the AWS Console process to create two separate Amazon ECR repositories: one named 'campus-bites-frontend' and another named 'campus-bites-backend'.
17. Provide the exact command string and explanation for authenticating my local Docker CLI with my AWS ECR registry using the 'aws ecr get-login-password' method.
18. Explain the series of Docker commands required to build a local image, tag it correctly for AWS ECR, and push it to a specific repository URI.
19. Detail the process of creating a secret within AWS Secrets Manager to securely store my PostgreSQL master credentials and connection endpoint.
20. Provide a guide for creating an IAM Task Execution Role. This role must have the specific managed policies required for ECS to pull images from ECR and retrieve secrets from Secrets Manager.
21. Explain how to create an ECS Task Role that allows my application code to interact with other AWS services, such as sending notifications via SNS or writing logs to CloudWatch.
22. Provide a deep dive into the hierarchy of Amazon ECS. Explain how a Cluster, Service, Task Definition, and Task relate to one another when using the serverless Fargate launch type.
23. Compare the ECS EC2 launch type against the AWS Fargate serverless launch type. Focus specifically on the cost implications and operational management overhead for a university-level student project.
24. Break down the components of an Amazon ECR image URI and explain a professional strategy for versioning images using semantic tagging (e.g., v1.0.1) instead of the 'latest' tag.
25. Explain the technical details of the 'awsvpc' networking mode in ECS Fargate. Why does it require an Elastic Network Interface (ENI) for every single running task?
26. Walk me through the creation of an ECS Cluster named 'CampusBites-Cluster' using the 'Networking-only' template specifically designed for AWS Fargate.
27. Provide a detailed JSON or Console-based guide for writing an ECS Task Definition for a React frontend. The configuration should use Fargate with 0.25 vCPU and 0.5 GB of RAM.

28. Explain how to define a container within an ECS Task Definition so that it pulls sensitive environment variables (like DB_PASSWORD) directly from AWS Secrets Manager at runtime.
29. Provide a technical explanation of how to map a container's internal port (e.g., port 80) to the host in a Fargate Task Definition to ensure the load balancer can reach the service.
30. Detail the configuration for a Node.js backend ECS Task Definition. This should be a Fargate task using 0.5 vCPU and 1.0 GB of RAM, listening on port 8080.
31. Provide the steps to create a Target Group for the External ALB. Ensure the target type is set to 'IP' mode and the traffic is directed toward the frontend tasks on port 80.
32. Explain how to configure advanced health checks for an ECS Target Group. Include settings for the /health endpoint, healthy/unhealthy thresholds, and the expected HTTP success codes.
33. Walk me through the deployment of an Internet-Facing Application Load Balancer. Ensure it is distributed across both of my public subnets for high availability.
34. Provide a guide on adding an HTTPS listener to an existing ALB. Explain how to associate a SSL/TLS certificate managed by AWS Certificate Manager (ACM) with the listener.
35. Detail the process for creating an Internal Application Load Balancer. This should be placed in my private app subnets to handle secure, internal API communication between tiers.
36. Explain how to create a Target Group for the Internal ALB. The targets must be the backend App tasks listening on port 8080, using the 'IP' target type.
37. Provide a step-by-step guide to creating an ECS Service for the Web Tier. It must maintain a desired count of 2 tasks and be integrated with the External ALB.
38. Explain the configuration requirements for an ECS Service using the 'awsvpc' network mode. Detail how to assign specific private subnets and the corresponding ECS Task Security Group.
39. Provide a guide to creating the App Tier ECS Service. Ensure it is connected to the Internal ALB and placed within the private app subnets for maximum isolation.

40. Explain the 'Force New Deployment' feature in ECS. How can I use this to ensure my service pulls the most recent image tag from ECR without changing the Task Definition?
41. Provide a technical walkthrough for setting up Service Auto Scaling for an ECS service. Configure it to trigger based on an average CPU utilization threshold of 70%.
42. Explain how to create a CloudWatch Alarm that monitors ECS service metrics and triggers a scale-out action (adding tasks) when CPU remains above 70% for three consecutive minutes.
43. Detail the configuration for a Scale-In policy for an ECS service. It should be designed to remove 1 task if the average CPU utilization stays below 30% for a period of 5 minutes.
44. Walk me through the process of setting up a Log Group in Amazon CloudWatch. How do I configure my Fargate containers to send their standard output (stdout) and error logs (stderr) to this group?
45. Provide a practical guide on using the Bastion Host. Explain how to establish an SSH tunnel from a local machine to test the connection to the private RDS PostgreSQL instance using the 'psql' utility.
46. Explain the verification process to ensure that the Internal ALB is correctly routing requests from the frontend Web tasks to the backend App tasks.
47. Provide a testing strategy to verify the High Availability of the architecture. How can I manually stop tasks in one Availability Zone to observe the load balancer and ECS service response?
48. Explain how to implement an Amazon ECR Lifecycle Policy. I need to automatically clean up the repository by deleting untagged images that are older than 14 days.
49. Provide a guide on using AWS Resource Groups. How can I use tags to logically group and organize every component of the 'CampusBites' project for easier management?
50. Explain how to verify that the NAT Gateway is functioning correctly. Provide a method to test outbound connectivity (like a simple curl command to an external API) from within a private app task.
51. Detail the steps to monitor 'Target Health' within the EC2 Console. What should I look for to ensure my Fargate tasks are correctly registered and passing health checks under the ALBs?

52. Explain the concept of 'Target Group Stickiness'. How do I enable this on an Application Load Balancer if my specific application needs to maintain persistent user sessions?
53. Provide a guide on modifying Security Group rules. Explain how to safely add a new inbound port rule to an existing group without disrupting current traffic or deleting existing rules.
54. Explain where to find the 'Stopped Reason' in the ECS Console when a task fails to start or crashes. Provide a list of common error codes and how to troubleshoot them.
55. Detail the process for setting up an Amazon SNS topic for infrastructure alerts. I want to receive email notifications if the RDS instance hits critical thresholds for storage or CPU usage.
56. Provide a technical guide on how to increase the allocated storage for an existing Amazon RDS instance. Explain how AWS handles this modification to minimize or avoid application downtime.
57. Explain how to use the AWS VPC Reachability Analyzer. Walk me through a test case to debug a connectivity failure between my Application Load Balancer and my ECS Fargate tasks.
58. Detail the benefits and configuration of cross-zone load balancing on an ALB. How does this ensure that traffic is distributed evenly across all healthy tasks regardless of their Availability Zone?
59. Provide a final teardown guide for the project. Explain how to create a final manual snapshot of the PostgreSQL database for archival purposes before deleting the RDS instance and associated resources.

Technical Report Prompts:

1. Provide a comprehensive Introduction for the 'CampusBites' Technical Report, detailing the shift from manual processes to a serverless 3-tier AWS architecture to mitigate canteen congestion.
2. Ok so Here is a thing, I want to dig up the references of the articles that have mentioned the services that I have used in this project, and I have found out the ai too that can do it for me, I want you to generate a prompt that can do the following:it

lists down all the services It tells the ai tool to dig up authentic and valid IEEE/APA articles and list their references.

3. Provide a detailed Problem Analysis section, focusing on temporal congestion during university breaks and the revenue leakages caused by unverified ordering systems.
4. Elaborate on the academic context of the project, citing research that validates container-based cloud pipelines and serverless efficiency for university-scale applications.
5. Create a Functional Requirements Traceability Matrix (Table 1) that maps user authentication, menu discovery, and kitchen dashboarding to specific AWS service components.
6. Detail the Non-Functional Requirements (Table 2), specifying the technical strategies used to achieve 99.9% availability, scalability, and network security through private subnets.
7. Provide an AWS Service Catalog section that describes the specific roles of VPC, IGW, and NAT Gateway in establishing the foundational network for the CampusBites system.
8. Explain the deployment of the External and Internal Application Load Balancers, detailing how they handle public traffic and private inter-service communication respectively.
9. Describe the Amazon ECS Cluster configuration, specifically detailing its role as the infrastructure boundary for containerized frontend and backend services.
10. Elaborate on the utilization of AWS Fargate as the serverless compute engine, highlighting how it removes the operational overhead of EC2 instance management.
11. Detail the purpose and configuration of the Bastion Host (t3.micro), explaining its role as a secure jump server for private database management.
12. Describe the Amazon ECR setup, focusing on how it manages versioned Docker images for the Node.js and React.js services to ensure a secure supply chain.
13. Provide a detailed breakdown of the 6-subnet VPC design (Table 4), listing the CIDR blocks and specific roles for public, app, and data subnets across two Availability Zones.

14. Explain the setup for Amazon RDS (PostgreSQL), specifying its placement in a private data subnet and the configuration of a DB Subnet Group for high durability.
15. Describe the integration of AWS Secrets Manager, detailing how ECS tasks securely retrieve PostgreSQL credentials and API keys at runtime via IAM roles.
16. Elaborate on the CloudWatch integration for monitoring CPU and Memory metrics to trigger automated scaling policies during peak break hours.
17. Detail the Compute Orchestration logic, explaining the separation of the Web and App services into distinct logical tiers within the ECS cluster.
18. Provide the specific ECS Fargate Task Configuration Parameters (Table 3), assigning vCPU and RAM allocations for both the frontend and backend jobs.
19. Describe the Network Topology, explaining how the use of private subnets and NAT Gateways ensures that backend tasks have no direct public exposure.
20. Explain the implementation of Administrative Access, focusing on the SSH tunneling process from the Bastion Host to the private PostgreSQL instance.
21. Detail the Image Management strategy in ECR, focusing on the support for rapid rollbacks and high-frequency deployment cycles in a serverless environment.
22. Provide a "Zero Trust" networking model explanation for the Security & Secrets Management section, detailing the logical flow from External ALB to RDS.
23. Elaborate on the Security Group Referencing Chain (Table 5), explaining why logical IDs are used instead of IP whitelisting to control traffic between tiers.
24. Detail the specific inbound and outbound rules for the Web-Server-SG, ensuring it only accepts traffic from the External-ALB-SG on port 80.
25. Describe the App-Server-SG ruleset, specifying that it only accepts API requests from the Internal-ALB-SG on port 8080.
26. Explain the RDS-SG configuration, detailing how it strictly limits PostgreSQL traffic to the App-Server-SG and the Bastion-SG for maintenance.
27. Detail the credential retrieval flow, explaining how the Node.js backend container queries AWS Secrets Manager upon startup using IAM Task Roles.
28. Provide an Operational Strategy section focused on Service Auto Scaling, explaining the reactive approach to handling "bursty" traffic patterns.

29. Elaborate on the Scale-Out Policy logic, specifying the CloudWatch triggers for adding tasks when average CPU exceeds a 70% threshold.
30. Describe the Scale-In Policy logic, explaining how the system removes tasks when utilization drops below 30% to optimize infrastructure costs.
31. Draft a Conclusion that summarizes the transition to a dual-ALB ECS Fargate architecture and its impact on university canteen efficiency and security.
32. Provide a comprehensive Reference list in IEEE format, including citations for cloud-native architectures, serverless efficiency, and distributed query privacy.
33. Explain the significance of the "awsvpc" network mode in ensuring that each Fargate task receives its own unique Elastic Network Interface (ENI).
34. Detail the health check configuration for the Application Load Balancer target groups, specifying the paths and thresholds for task recycling.
35. Describe the Multi-AZ deployment strategy for RDS, explaining the benefit of synchronous standby replicas for data persistence.
36. Elaborate on the role of IAM Task Execution Roles in granting ECS the permissions to pull Docker images and write logs to CloudWatch.
37. Draft an Executive Summary for the CampusBites proposal, outlining the 3-tier serverless container strategy for university canteen modernization.
38. Provide a detailed Problem Statement focusing on severe physical congestion and revenue loss due to anonymous ordering in manual cafeterias.
39. Elaborate on the operational inefficiencies of the current manual model, emphasizing the need for staff to refocus on food preparation rather than admin tasks.
40. Detail the Functional Requirements for the proposal, including mandatory login via Student ID and real-time menu browsing with availability checks.
41. Describe the Kitchen Dashboard requirement, explaining how it provides staff with a real-time interface to manage the incoming order queue.
42. Specify the Non-Functional Requirements, focusing on high availability across two Availability Zones and maintainability via serverless Fargate tasks.
43. Provide a detailed AWS Cloud Architecture breakdown, listing every service from ECR for image storage to RDS for PostgreSQL data management.

44. Describe the role of the ECS Cluster as the infrastructure boundary and ECS Services as the orchestrators for the container lifecycle.
45. Explain the utility of the NAT Gateway in enabling private app tasks to fetch software updates without public internet exposure.
46. Detail the Deployment Strategy for the Web Tier, explaining its placement in public subnets to handle internet-facing traffic via the ALB.
47. Describe the App Tier deployment in private subnets, ensuring it is strictly isolated and only accessible via the internal load balancer.
48. Elaborate on the Data Tier isolation, specifying that the PostgreSQL RDS instance accepts traffic solely from the backend application logic.
49. Explain the Security & Secrets Management strategy, focusing on the use of IAM Roles to retrieve database credentials from AWS Secrets Manager at runtime.
50. Detail the Administrative Access requirements, specifically the provisioning of a Bastion Host in a public subnet for secure database management.
51. Provide a conclusion that highlights how the ECS Fargate transition ensures a modern, scalable, and secure platform for university students.
52. Describe the accountability measures implemented through identity verification, ensuring every order is linked to a verified Student or Staff ID.
53. Explain the dynamic scaling requirements for handling the 15-30 minute break windows where cafeteria demand peaks.
54. Summarize the technical benefits of using Docker containers for the frontend and backend to ensure environment parity between development and production.

PPT Prompts:

For PPT we just gave gama.ai (tool for making presentations) our technical report and it made the presentation for us.