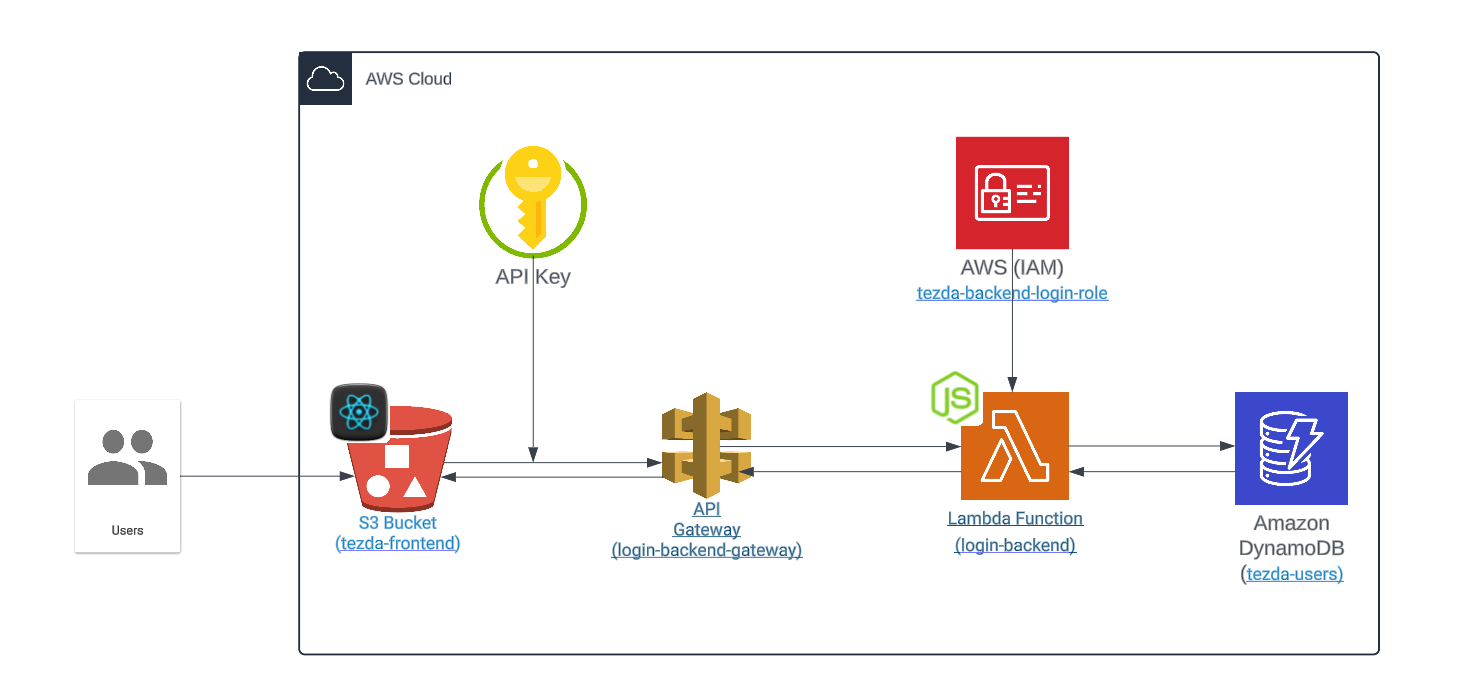
**Complete Architecture of The Project: -**



**Security Measures Implemented:**

1. **Input validation for username, email & password:** This ensures that the data entered by users meets certain criteria, preventing malicious input.
2. **Input sanitization to avoid basic XSS attack:** By sanitizing input, you mitigate the risk of cross-site scripting (XSS) attacks, which could otherwise manipulate the behaviour of the application.
3. **Storing encrypted passwords using hashing technique:** Storing passwords in encrypted form with a strong hashing technique adds an extra layer of security, preventing exposure of plain text passwords in case of a data breach.
4. **Implemented JSON Web Token (JWT) for authorized requests:** JWTs provide a secure way to transmit information between parties, ensuring that only authorized users can access certain resources.
5. **Monitoring & logging enabled in AWS via CloudWatch:** This allows for real-time monitoring of system behaviour and logs important events for auditing and troubleshooting purposes.
6. **Enabled “API Key” required for every endpoint:** API keys provide a way to authenticate and control access to APIs, preventing unauthorized access.
7. **Implemented rate limiting using “usage plan” in API Gateway:** Rate limiting helps prevent abuse of API resources by limiting the number of requests a client can make within a certain time frame.
8. **Created a specific IAM role for the lambda function:** Restricting permissions through IAM roles ensures that your Lambda function can only interact with specific AWS services, reducing the attack surface.

**Potential Security Improvements:**

1. **Enabling HTTPS with SSL/TLS certificate:** This is a crucial step for securing data transmission between users and Lambda function. Without HTTPS, data can be intercepted and tampered with during transit. Enabling HTTPS ensures data integrity and confidentiality.
2. **Implementing Two-Factor Authentication (2FA):** Enhance user authentication security by implementing 2FA, which adds an additional layer of verification beyond passwords, such as using SMS codes, authenticator apps, or biometric authentication.

**Resource Utilization and Cost-Effectiveness:**

1. **Right-sizing Lambda resources:** Ensure that the allocated memory and CPU resources for your Lambda function are optimized for its workload. Over-provisioning can lead to unnecessary costs, while under-provisioning can affect performance.
2. **Performance monitoring and optimization:** Continuously monitoring the performance of Lambda function and identifying any bottlenecks or areas for optimization. Utilize AWS CloudWatch metrics and logs to gain insights into function execution times, resource utilization, and error rates.

I have completed the implementation of the register and login functionality as per the assignment requirements. You can review the code attached and visit the website to see it in action.

**Website URL**: <http://tezda-frontend.s3-website.ap-south-1.amazonaws.com>