Cloud Cost Optimization Report – Student Management Website

SECTION 1: Project Overview

The Student Management Website is a web application designed to manage student data, accessible by students, parents, faculties, and college administration. The project aims to handle up to 500 users on a busy day, ensuring high scalability, consistent and reliable data handling, and accurate updates. The primary business goal is to provide a robust and efficient platform for managing student information, enhancing communication and administrative processes. Technically, the project will leverage React for the frontend, Ruby on Rails for the backend, and MySQL for the database, with a monthly budget of INR 20,000.

SECTION 2: Project Architecture & Requirements

The key components of the project include:
- Frontend: React for a dynamic and responsive user interface.
- Backend: Ruby on Rails for handling business logic and API endpoints.
- Database: MySQL for storing and managing student data.
- Non-functional Requirements:
- Highly Scalable Architecture: The system must be able to handle increasing loads and user traffic.
- Consistent and Reliable Data Handling: Data integrity and accuracy are paramount.
- Slow Updates Acceptable: While updates can be slow, they must always be correct.
- Expected Scale: The application is designed to handle up to 500 concurrent users on a busy day, with the potential for future growth.

SECTION 3: Billing Overview

The project utilizes multiple AWS services, each contributing to the overall cost. The billing scenarios include:
- EC2: Web server for hosting the React frontend and Ruby on Rails backend.
- RDS: MySQL database for storing student data.
- S3: Storage for static assets and backups.
- CloudFront: Content delivery network for faster content delivery.
- Route 53: DNS management for the website.
- VPC: Network traffic monitoring.
- IAM: Identity and access management.
- CloudWatch: Monitoring and logging.
- Lambda: Serverless functions for background tasks.
- SNS: Notification service for system alerts.
- SES: Email service for user notifications.
- Elasticache: In-memory data store for caching.
- ElasticBeanstalk: Platform as a service for deploying the application.
- CloudFormation: Infrastructure as code for resource management.
- CodePipeline: Continuous integration and delivery pipeline.
- CodeBuild: Build service for compiling and testing code.
- CodeDeploy: Deployment service for rolling out updates.
- XRay: Distributed tracing for debugging and performance analysis.

The cost ranges and variability are influenced by the usage of these services, with some services having fixed costs and others varying based on usage.

SECTION 4: Cost Analysis

- Total Monthly Cost: INR 6,850
- Budget: INR 20,000
- Budget Variance: INR 13,150 (under budget)
- Key Cost Drivers:
- AWS - RDS: INR 1,500
- AWS - EC2: INR 1,200
- AWS - ElasticBeanstalk: INR 800
- AWS - CloudFront: INR 500
- AWS - Elasticache: INR 600
- High-Cost Services:
- RDS: INR 1,500
- EC2: INR 1,200
- ElasticBeanstalk: INR 800
- CloudFront: INR 500
- Elasticache: INR 600
- Overall Cost Efficiency Assessment: The current cost is significantly under the budget, indicating efficient resource utilization. However, there are opportunities to further optimize costs and improve efficiency.

SECTION 5: Optimization Recommendations

- Optimize EC2 Instance Usage
- Service: AWS - EC2
- Current Cost: INR 1,200
- Potential Savings: INR 300
- Recommendation Type: Right-sizing
- Description: Evaluate the current EC2 instance type and consider using a smaller instance type or spot instances to reduce costs.
- Implementation Effort: Medium
- Risk Level: Low
- Steps:
- Review current EC2 instance usage and performance metrics.
- Test with a smaller instance type or spot instances.
- Monitor performance and adjust as needed.
- Cloud Providers: AWS

- Optimize RDS Instance Usage
- Service: AWS - RDS
- Current Cost: INR 1,500
- Potential Savings: INR 450
- Recommendation Type: Right-sizing
- Description: Evaluate the current RDS instance type and consider using a smaller instance type or read replicas to reduce costs.
- Implementation Effort: Medium
- Risk Level: Low
- Steps:
- Review current RDS instance usage and performance metrics.
- Test with a smaller instance type or read replicas.
- Monitor performance and adjust as needed.
- Cloud Providers: AWS

- Optimize CloudFront Usage
- Service: AWS - CloudFront
- Current Cost: INR 500
- Potential Savings: INR 150
- Recommendation Type: Optimization
- Description: Optimize CloudFront by enabling caching for static assets and using a smaller distribution size.
- Implementation Effort: Low
- Risk Level: Low
- Steps:
- Enable caching for static assets.
- Review and optimize CloudFront distribution settings.
- Cloud Providers: AWS

- Optimize Elasticache Usage
- Service: AWS - Elasticache
- Current Cost: INR 600
- Potential Savings: INR 180
- Recommendation Type: Right-sizing

- Description: Evaluate the current Elasticache instance type and consider using a smaller instance type or optimizing cache usage.
- Implementation Effort: Medium
- Risk Level: Low
- Steps:
- Review current Elasticache usage and performance metrics.
- Test with a smaller instance type or optimize cache usage.
- Monitor performance and adjust as needed.
- Cloud Providers: AWS

- Optimize ElasticBeanstalk Usage
- Service: AWS - ElasticBeanstalk
- Current Cost: INR 800
- Potential Savings: INR 240
- Recommendation Type: Optimization
- Description: Optimize ElasticBeanstalk by using auto-scaling policies and adjusting environment settings to better match usage patterns.
- Implementation Effort: Medium
- Risk Level: Low
- Steps:
- Review current ElasticBeanstalk environment settings.
- Implement auto-scaling policies.
- Monitor performance and adjust as needed.
- Cloud Providers: AWS

- Use Open-Source Alternatives for Monitoring
- Service: AWS - CloudWatch
- Current Cost: INR 200
- Potential Savings: INR 150
- Recommendation Type: Alternative Provider
- Description: Consider using open-source monitoring tools like Prometheus and Grafana to reduce CloudWatch costs.
- Implementation Effort: High
- Risk Level: Medium
- Steps:
- Evaluate open-source monitoring tools.
- Set up and configure Prometheus and Grafana.
- Monitor and adjust configurations as needed.
- Cloud Providers: Open-source

- Optimize S3 Storage
- Service: AWS - S3
- Current Cost: INR 200
- Potential Savings: INR 50
- Recommendation Type: Optimization
- Description: Optimize S3 storage by using lifecycle policies to move infrequently accessed data to cheaper storage classes.
- Implementation Effort: Low
- Risk Level: Low
- Steps:

- Review current S3 storage usage.
- Set up lifecycle policies to move data to cheaper storage classes.
- Cloud Providers: AWS

- Optimize Lambda Usage
- Service: AWS - Lambda
- Current Cost: INR 400
- Potential Savings: INR 120
- Recommendation Type: Optimization
- Description: Optimize Lambda functions by reducing the memory allocation and improving function efficiency.
- Implementation Effort: Medium
- Risk Level: Low
- Steps:
- Review current Lambda function usage and performance metrics.
- Optimize function code and reduce memory allocation.
- Monitor performance and adjust as needed.
- Cloud Providers: AWS

SECTION 6: Estimated Savings & Impact

- Key Components:
- EC2 Instance Usage
- RDS Instance Usage
- CloudFront Usage
- Elasticache Usage
- ElasticBeanstalk Usage
- CloudWatch Monitoring
- S3 Storage
- Lambda Usage
- Total Potential Savings: INR 1,640
- Percentage Savings: 23.94%
- Business Impact: Implementing these optimizations will not only reduce costs but also enhance the performance and reliability of the Student Management Website. The savings can be reinvested in additional features or used to further optimize the application.

SECTION 7: Final Summary & Next Steps

- Key Takeaways:
- The current cost is significantly under the budget, indicating efficient resource utilization.
- There are several opportunities to further optimize costs and improve efficiency.
- The top cost drivers are RDS, EC2, ElasticBeanstalk, CloudFront, and Elasticache.
- What Should Be Implemented First:
- Optimize EC2 and RDS instance usage, as they are the highest cost drivers.
- Implement CloudFront and Elasticache optimizations to reduce content delivery and caching costs.
- Long-Term Optimization Strategy:
- Continuously monitor and review resource usage and performance metrics.
- Regularly evaluate and implement new optimization techniques and tools.
- Consider using cost management tools and services to automate cost optimization processes.
- Stay updated with the latest AWS offerings and best practices for cost optimization.