Requirement Document: FinOps Application for AWS Cost Optimization

# 1. Objective

Develop a web-based application that enables AWS cost and resource optimization by leveraging AWS Trusted Advisor APIs. The application will display real-time Trusted Advisor recommendations and allow automated remediation actions to implement those recommendations.

# 2. Scope

## In-Scope

- Fetching AWS Trusted Advisor recommendations using boto3  
- Displaying categorized recommendations in a Vue.js frontend  
- Backend implementation using FastAPI or Flask  
- Executing automated remediation actions  
- Secure authentication and logging  
- Deployment on AWS

## Out-of-Scope (for MVP)

- Multi-cloud FinOps support (GCP, Azure)  
- Role-based access control with fine-grained policies  
- Custom recommendation engine  
- Historical trend analytics dashboard

# 3. Key Features

|  |  |
| --- | --- |
| **Feature** | **Description** |
| View Trusted Advisor Checks | List checks by category (cost, security, fault tolerance, performance). |
| View Recommendations | Show insights with affected resources. |
| Implement Button | Trigger automation to apply recommended changes. |
| Real-Time Feedback | Notify user of successful or failed implementations. |
| Authentication | AWS IAM or Cognito-based login. |
| Logging and Auditing | Capture action history for compliance. |

# 4. Assumptions

- AWS account has Business or Enterprise support plan  
- Appropriate IAM permissions are set  
- Automation logic will cover only low-risk, reversible actions

# 5. Constraints

- AWS Trusted Advisor API has rate limits  
- Some recommendations may require manual confirmation  
- Remediations must be idempotent and safe

# 6. Benefits of FinOps Application

|  |  |
| --- | --- |
| **Area** | **Benefit** |
| Cost Optimization | Automatically reduces unused resources |
| Operational Efficiency | Reduces manual effort |
| Governance | Provides transparency and controlled access |
| Scalability | Extendable to multiple AWS accounts |
| Security Posture | Improves compliance and fixes vulnerabilities |

# 7. Work Breakdown Structure (WBS)

A detailed WBS organized by major deliverables and task owners. Tasks are grouped under backend, frontend, devops, QA, and documentation categories.

## 7.1 Project Setup

| **ID** | **Task** | **Owner** |
| --- | --- | --- |
| 1.1 | Define AWS accounts and permissions | Architect |
| 1.2 | Setup GitHub repo / CI-CD pipeline | DevOps |
| 1.3 | Define coding standards and structure | Architect |

## 7.2 Backend Development (FastAPI or Flask)

| **ID** | **Task** | **Owner** |
| --- | --- | --- |
| 2.1 | Setup FastAPI base project structure | Backend Dev |
| 2.2 | Integrate boto3 for Trusted Advisor | Backend Dev |
| 2.3 | GET /recommendations endpoint | Backend Dev |
| 2.4 | POST /implement/<check\_id> endpoint | Backend Dev |
| 2.5 | Implement 3–5 automation scripts (EC2, EBS, S3, etc.) | Backend Dev |
| 2.6 | Add logging, error handling, retry logic | Backend Dev |
| 2.7 | Create Swagger/OpenAPI docs | Backend Dev |

## 7.3 Frontend Development (Vue.js)

| **ID** | **Task** | **Owner** |
| --- | --- | --- |
| 3.1 | Initialize Vue 3 project | Frontend Dev |
| 3.2 | Build RecommendationList.vue | Frontend Dev |
| 3.3 | Build RecommendationItem.vue | Frontend Dev |
| 3.4 | Axios integration with backend | Frontend Dev |
| 3.5 | Implement notification and feedback UI | Frontend Dev |
| 3.6 | Basic styling and responsiveness | Frontend Dev |

## 7.4 Authentication and Security

| **ID** | **Task** | **Owner** |
| --- | --- | --- |
| 4.1 | Setup AWS Cognito for authentication | DevOps |
| 4.2 | Secure backend endpoints with IAM tokens | Backend Dev |
| 4.3 | Frontend login / token-based auth | Frontend Dev |
| 4.4 | Role-based action restrictions (basic) | Architect |

## 7.5 Testing and QA

| **ID** | **Task** | **Owner** |
| --- | --- | --- |
| 5.1 | Unit tests for backend APIs | QA / Backend Dev |
| 5.2 | Unit tests for Vue components | QA / Frontend Dev |
| 5.3 | Integration testing with mocked AWS calls | QA |
| 5.4 | UAT testing with real AWS account | QA |

## 7.6 Deployment

| **ID** | **Task** | **Owner** |
| --- | --- | --- |
| 6.1 | Deploy frontend on S3 or Amplify | DevOps |
| 6.2 | Deploy backend on Lambda/Fargate/EC2 | DevOps |
| 6.3 | Setup logging (CloudWatch / custom logs) | DevOps |
| 6.4 | Setup alarms and error notification | DevOps |

## 7.7 Documentation and handover

| **ID** | **Task** | **Owner** |
| --- | --- | --- |
| 7.1 | Developer documentation | All |
| 7.2 | User guide / admin panel training | Architect |
| 7.3 | Final code handover and walkthrough | All |

# 8. Estimated Timeline (Agile Approach)

|  |  |  |
| --- | --- | --- |
| Sprint | Tasks | Duration |
| Sprint 1 | Backend + Frontend scaffold, TA API integration | 2 weeks |
| Sprint 2 | UI + Automation scripts | 2 weeks |
| Sprint 3 | Auth, Logging, Testing | 2 weeks |
| Sprint 4 | UAT, Documentation, Deployment | 2 weeks |