**Project Proposal: Secure Personal Finance Dashboard on AWS Free Tier**

Paulette Wolfe

Department of Data Science Monroe University, King Graduate School

25SP-CS670-152HY: Cloud Computing

Professor: Mahmud Islam

June 17, 2025

**1. Project Overview**

This project aims to design and deploy a secure, serverless **Personal Finance Dashboard** that enables users to track income, expenses, and financial goals via a modern web interface. Built entirely within the AWS Free Tier, it emphasizes cloud-native architecture, user-centric design, and security by design.

**2. Objectives**

* Create a secure, responsive dashboard for personal financial tracking.
* Implement secure backend APIs to interact with financial data.
* Ensure proper authentication, data encryption, and access control across all components.
* Stay within AWS Free Tier limits for cost-efficiency during development.

**3. AWS Architecture and Services**

| **Purpose** | **AWS Service** | **Free Tier Details** |
| --- | --- | --- |
| Web Hosting | **Amazon S3 + CloudFront** | 5 GB storage, 20K GET, 2K PUT requests |
| Backend API | **Amazon API Gateway** | 1M REST API calls/month |
|  | **AWS Lambda** | 1M requests and 400,000 GB-sec compute/month |
| Data Storage | **Amazon DynamoDB** | 25 GB storage, 200M requests/month |
| Visualization (Optional) | **Amazon QuickSight** | 1 user, SPICE capacity free for 60 days |

**4. Project Scope (with Integrated Security)**

**Frontend (Web Interface via Amazon S3 + CloudFront)**

* Static website hosting with HTML/CSS/JavaScript.
* Hosted on S3 with CloudFront as CDN for secure, fast delivery.
* **Security Controls:**
  + S3 public access blocked; use CloudFront Origin Access Identity (OAI) to control access.
  + Enable S3 server-side encryption (SSE-S3 or SSE-KMS).
  + Enable S3 access logging for audit visibility.

**Backend (Serverless API with API Gateway + AWS Lambda)**

* RESTful API endpoints (e.g., /addTransaction, /getSummary).
* Lambda functions handle all business logic.
* **Security Controls:**
  + Use IAM roles or Amazon Cognito user pools for authentication and authorization.
  + Use API keys with usage plans and throttling to prevent abuse.
  + Enforce HTTPS-only via API Gateway.
  + Use least-privilege IAM policies for Lambda execution roles.
  + Secure environment variables; store secrets in AWS Secrets Manager.
  + Validate and sanitize all inputs to Lambda to prevent injection attacks.
  + Optionally integrate AWS WAF to protect against common web threats.

**Data Layer (Amazon DynamoDB)**

* NoSQL table to store financial transactions (userId, amount, category, date, type).
* **Security Controls:**
  + Use IAM-based access policies to control table access by role/user.
  + Enable encryption at rest using AWS-managed KMS.
  + Enable Point-in-Time Recovery (PITR) for data protection.
  + Interact via HTTPS to prevent data interception.

**Visualization (Amazon QuickSight)**

* Connect to DynamoDB via SPICE or ETL pipeline.
* Provide visuals such as spending trends, income vs expenses, etc.
* Security Controls:
  + Enable SSO or IAM-based authentication.
  + Limit access to dashboards per user role.
  + Avoid public sharing of dashboards or datasets.

**5. Benefits and Justification**

* Secure by design: All services implement authentication, encryption, access control, and logging.
* Cost-effective: Fully utilizes AWS Free Tier resources.
* Scalable: Built with serverless services that scale automatically.
* Low maintenance: Managed services reduce operational overhead.
* Extensible: Designed for future integration with mobile apps, budgeting tools, or financial APIs.

**6. Timeline**

| **Week** | **Milestone** |
| --- | --- |
| 1 | Requirements gathering, secure S3 static hosting with CloudFront |
| 2 | Build backend APIs using secure Lambda + API Gateway |
| 3 | DynamoDB schema creation + IAM access policies |
| 4 | Frontend integration with secure backend |
| 5 | Optional: QuickSight dashboard and access setup |
| 6 | Testing, security audit, and performance optimization |

**7. Deliverables**

* Securely hosted, browser-accessible Personal Finance Dashboard
* RESTful API documentation with security details
* DynamoDB data schema with access controls
* Deployment and security guide (IAM roles, policies, usage limits)

**8. Risks and Mitigation**

| **Risk** | **Mitigation** |
| --- | --- |
| Data exposure via public endpoints | Enforce IAM, Cognito auth, HTTPS, and API throttling |
| IAM misconfiguration | Follow **the least privilege principle** and test with temporary policies |
| Free Tier limits exceeded | Set **AWS Budget alerts** and monitor via **CloudWatch + CloudTrail** |
| Secret/key exposure | Use **Secrets Manager**, avoid secrets in code or environment vars |

**9. Conclusion**

This secure AWS Personal Finance Dashboard leverages the Free Tier to deliver meaningful insights with zero upfront costs. By integrating security from the start via encryption, IAM, network protection, and AWS best practices. It provides a resilient, production-ready foundation for personal finance management or future SaaS development.