University of Phoenix®

Project Plan Draft

Fill out each of the sections below with information relevant to your project. Be sure to include the company name associated with your project.

Company Name:

Network Technology Recommendations

Network Technology Selection Criteria

Selection Criteria Name ***(short name to ID the criteria)***	**Selection Criteria Description** ***(Define the criteria for technology to associate with a point value.)***	**Selection Criteria Value (Weighting in Points)** ***(e.g., 3 – Excellent, 2 – Good, 1 – Acceptable)***
Scalability	Ability to handle 10,000+ concurrent connections and scale on demand	3
Security	Multi-layer security, encryption, isolation (private subnets, IAM)	3
Performance	Low-latency response for global users; minimized network hops	2
Manageability	Ease of configuration and maintenance (automation, infrastructure-as-code)	2
Cost Effectiveness	Overall TCO (infrastructure, maintenance)	2

Network Technology Recommendation

Recommended Network Technologies	**Description**	**Benefits**	**Aggregate Selection Criteria Score** ***(Score for this technology based on the selection criteria detailed above.)***
Multi-Tier Hub-	A layered VPC architecture	- **High Scalability**: Quickly add	3 + 3 + 3 = 9

and-Spoke on AWS VPC	with public subnets (load balancers) and private subnets (application, DB). Leverages AWS application load balancer, Security groups, and VPC peering	subnets or replicate them in multiple regions. - **Strong Security**: Private subnets, IAM, and security groups protect backend resources. - **Manageable**: Infrastructure-as-code, plus native AWS tools for monitoring and automation.	
Azure Virtual Network with Hub-and-Spoke Architecture	This design centralizes connectivity and routing, making it easier to manage traffic between multiple subnets and regions.	- **Scalability:** Easily expands by adding new spokes or regions as your demand grows. - **Security:** Utilizes built-in Network Security Groups (NSGs) and Azure Firewall to enforce security policies and isolate sensitive resources. - **Manageability:** Centralized management via the Azure Portal and automation through ARM templates simplifies configuration and ongoing maintenance.	3 + 3 + 2 = 8

Network Technology Vendor Selection Criteria

(*Third-party technology provider*)

Selection Criteria Name	Selection Criteria Description	Selection Criteria Value (Weighting in Points)
Trusted in Tech	Vendor must have extensive experience with enterprise-scale cloud networking	3
Security & Compliance	Must offer compliance with banking/financial industry standards (PCI DSS, SOC 2, etc.)	3
Global Presence	Data centers or PoPs worldwide for low-latency and redundancy	2
Integration	Seamless integration with AWS services (VPC, Route 53, CloudFront, WAF)	3

Network Technology Recommended Vendors

Vendor Name	Vendor Strengths	Vendor Weaknesses	Products/Services Provided to Project	Aggregate Selection Criteria Score
Amazon Web Services (AWS)	- AWS VPC integration - Strong security and compliance - Globally available	- Proprietary environment - Cost can rise at scale	- AWS VPC, Subnets, NAT Gateways - AWS Route 53, WAF, CloudFront - Security Groups, Network ACLs	3 + 3 + 3 = 9
Microsoft Azure	- Seamless integration with Azure Virtual Network - Enterprise-grade security and compliance certifications - global data centers with robust hybrid cloud support	- Licensing and management complexity - Costs may increase with scale	- Azure Virtual Network, Subnets, and VPN Gateway - Azure ExpressRoute, Azure Firewall, and Traffic Manager - Network Security Groups (NSGs) and Route Tables	3 + 3 + 2 = 8

Network Technology Deployment Challenges

Deployment Challenge *(short name to ID the challenge)*	Deployment Challenge Description *(What obstacles can potentially complicate or delay deployment of technology, and affect the project timeline?)*
Network Complexity	Correctly configuring subnets, routing tables, and security groups to ensure no unauthorized access but still allow internal traffic can be quite challenging
Global Latency	Ensuring minimal latency for users worldwide—may require multi-region strategy or CloudFront caching
Compliance Audits	Rigorous documentation and proof that the network meets banking/financial compliance standards (PCI DSS, SOC 2, etc.)
Improvement #1	Implement AWS Config and AWS Security Hub to automatically monitor compliance with

	financial regulations (PCI DSS, SOC 2). (This change is based on research highlighting the importance of automated compliance tracking in finance to reduce audit failure risks and support real-time visibility.)

Technology Adoption Methods

Method Name ***(short name to ID the method)***	**Method Description** ***(Summarize the process for adopting the technology.)***
Infrastructure-as-Code (IaC)	Use AWS CloudFormation or Terraform to define all subnets, security groups, routing, etc. in versioned templates, enabling repeatable, controlled deployments
Pilot / Test Environment	Deploy a smaller-scale pilot environment in a non-production account, validate latency, security configurations, then replicate for production

Cost/Benefit Considerations

Benefits	**Costs**	**Considerations**
Simplifies adding new regions or subnets	AWS service charges for NAT Gateways, data transfer, NAT traffic	Must factor in egress data transfer for global users
Centralized security controls	Price increases with scaling	Possibly more complex to configure at first
Automatic scaling and load balancing	Price increases with scaling	Plan for multi-AZ usage to ensure high availability

Database System Recommendation

Database System Selection Criteria

Selection Criteria Name	**Selection Criteria Description**	**Selection Criteria Value (Weighting in Points)**
ACID Compliance	Must support atomic, consistent, isolated, durable transactions	3
Scalability	Ability to scale read/write operations (horizontal read replicas, vertical scaling)	2

Security & Compliance	Encryption at rest, encryption in transit, auditing, user access controls, meets financial data standards	3
Performanc e	Low latency for queries and transactions (under 2 seconds for 95% of operations)	2

Database System Recommendation

Recommended Database System	Description	Benefits	Aggregate Selection Criteria Score
Amazon RDS (MySQL or PostgreSQL) (Relational)	Managed relational database service offering built-in security features, automated backups, read replicas, and multi-AZ failover. Supports ACID transactions critical for finance.	- Acid Compliance - Automatic Scaling - Automated Backups - Built-in Security	3
Azure SQL Database	A fully managed relational database service that supports ACID transactions, automated backups, and high availability. It's designed to scale dynamically with your application's needs while ensuring robust security and compliance for financial data.	- Acid Compliance - Automatic Scaling - Automated Backups - Built-in Security	3

Database System Vendor Selection Criteria

Selection Criteria Name	Selection Criteria Description	Selection Criteria Value (Weighting in Points)
Reliability & Uptime	Vendor should offer 99.99% or higher SLA	3
Automated Management	Automated patching, backups, and failover	2
Scalability Options	Support for read replicas, or horizontal scale solutions	2
Security & Compliance Features	Must provide encryption, auditing, and adhere to financial compliance	3

Database System Recommended Vendors

Vendor Name	Vendor Strengths	Vendor Weaknesses	Products/Services Provided to Project	Aggregate Selection Criteria Score
Amazon Web Services (RDS	- Automated Backups - High availability - Encryption in transit/rest	- Proprietary hosting - Additional cost for read replicas	Amazon RDS for MySQL/PostgreSQ	3
Microsoft Azure SQL DB	managed service with automated backups and high availability - Built-in encryption, auditing, and compliance features - Seamless integration with other Azure services (e.g., monitoring, scaling, analytics)	- Proprietary platform that can lead to vendor lock-in - Potentially higher costs at scale, especially with advanced features - Newer AI features may be complex to use	Azure SQL Database (Managed relational database service supporting ACID transactions, geo-replication, and high performance)	2

Database System Deployment Challenges

Deployment Challenge	Deployment Challenge Description
Migration of Data	Importing data from legacy systems or other systems may come with problems
Compliance Auditing	Setting up logs, encryption keys, and audits to prove compliance
Ensuring ACID on High Load	Handling spikes in transactions without losing performance

Technology Adoption Methods

Method Name	Method Description
Proof of Concept (PoC)	Start with a smaller RDS instance in a dev environment to test queries, performance, and security before scaling to production
Database as Code	Manage DB configurations using AWS CloudFormation or Terraform, ensuring consistent dev/stage/prod environments

Cost/Benefit Considerations

Benefits	Costs	Considerations
Fully Managed (backups and patching)	Pay-as-you-go for instances, storage, I/O	Must plan for future capacity growth
Scalable read replicas	Additional cost for Multi-AZ support	Need to evaluate read/write patterns for cost optimization
Strong security & compliance	Additional cost for advanced features	Maintain patch schedules that align with uptime/service windows

Software Application Recommendations

Software Application Selection Criteria

Selection Criteria Name	Selection Criteria Description	Selection Criteria Value (Weighting in Points)
Modern Web Framework	Must support server-side rendering, fast builds, easy dev experience	3
Security and Auth	Ability to integrate multi-factor auth, secure sessions, and user role management	3
Performance Optimization	Minimizing page load times globally (CDN integration, code splitting, caching)	2
Developer Productivity	Clear documentation, strong community, robust tooling	2

Software Application Recommendation

Recommended Software Application	Description	Benefits	Aggregate Selection Criteria Score

Next.js (Frontend + Backend)	React-based framework for SSR, SSG, and API routes. Facilitates secure, rapid dev, dynamic pages, and easy scaling with serverless or container-based hosting	- **Modern Web Framework**: Developer-friendly, SSR/SSG for performance. - **Security**: Integrates easily with OAuth, multi-factor auth libraries. - **Performance**: Code splitting, caching, works well with CDN. - **Productivity**: Large community, official docs, plugin ecosystem.	3 + 3 + 3 + 9 = 12
Angular	A server-side rendering solution for Angular applications.	- **Modern Web Framework:** Provides a full-featured, opinionated framework with strong modular architecture and two-way data binding. - **Security:** Incorporates built-in security best practices and leverages TypeScript for robust, maintainable code, enhanced by Angular Universal for improved SEO and server-side performance. - **Performance:** Employs Ahead-of-Time (AOT) compilation, efficient change detection, and tree-shaking to optimize bundle sizes and loading times. - **Productivity:** Features an integrated CLI, extensive official documentation, and a rich ecosystem of libraries and tools that streamline development.	3 + 3 + 3 + 2 = 11
Nuxt.js	A framework built on Vue.js that offers server-side rendering, static site generation, and a streamlined development experience.	- **Modern Web Framework:** Seamlessly extends Vue.js with built-in support for SSR and static site generation, offering a balanced mix of flexibility and simplicity. - **Security:** Leverages Vue's reactive architecture alongside ecosystem tools for authentication and	3 + 2 + 2 + 3 = 10

		authorization to build secure applications. - **Performance:** Automatically handles code splitting, optimizes bundle sizes, and prefetches resources to ensure fast page loads and responsive interfaces. - **Productivity:** Boasts a robust module ecosystem, intuitive configuration, and strong community support that accelerates development and maintenance.	

Software Application Vendor Selection Criteria

Selection Criteria Name	Selection Criteria Description	Selection Criteria Value (Weighting in Points)
Active Community & Support	Must have extensive community support, frequent updates, and official documentation	3
Production Use Cases	Framework proven in large-scale production apps with strong performance	3
Integration with AWS	Must easily integrate with AWS serverless or container hosting	2

Software Application Recommended Vendors

Vendor Name	Vendor Strengths	Vendor Weaknesses	Products/Services Provided to Project	Aggregate Selection Criteria Score
Vercel (Creators of Next.js)	- Official maintainers of Next.js - Advanced edge network solutions - One-click deployments, strong dev tooling	- May incur higher costs at scale - Some advanced features proprietary	- Next.js core framework - Possibly serverless Edge Function	3 + 2 = 5
Open-	- Completely	- No official	- Next.js framework itself	2

Source Next.js (OpenNext)	free to use - Large, active GitHub community	“enterprise” support		

Software Application Deployment Challenges

Deployment Challenge	Deployment Challenge Description
SSR & Serverless Integration	Large Next.js bundles might exceed Lambda code package size—may require container-based or specialized serverless framework
Authentication & Session Handling	Must ensure secure session tokens or OAuth flows that meet banking security standards
Rapid Release Cycles	Frequent iteration demands robust CI/CD to avoid downtime or regression

Technology Adoption Methods

Method Name	Method Description
Agile Sprints	Break development into sprints (1-2 weeks). Each sprint includes Next.js feature dev, testing, stakeholder review
CI/CD Integration	Use AWS CodePipeline or GitHub Actions to automate building and deploying Next.js code on each commit

Cost/Benefit Considerations

Benefits	Costs	Considerations
Rapid dev cycle with Next.js	Development team wages	Plan for code optimization (tree shaking, minification)
Strong performance via SSR & caching	Learning curve for server components	Incorporate security best practices
Compatible with AWS	DevOps integration	Ensure ongoing alignment between Next.js releases and AWS infrastructure updates

Cloud Services Recommendations

Cloud Services Selection Criteria

Selection Criteria Name	Selection Criteria Description	Selection Criteria Value (Weighting in Points)
Scalability & Reliability	Must automatically handle large traffic spikes (10,000+ concurrent users) and deliver 99.99% uptime	3
Global Distribution	Ability to serve content at low latency worldwide (CDN, multi-region)	3
Security & Compliance	Built-in compliance with banking standards, encryption, and identity management	3
Integration with Next.j	Seamless or minimal-friction deployment for SSR/SSG or container-based Next.js apps	2

Cloud Services Recommendation

Recommended Software Application	Description	Benefits	Aggregate Selection Criteria Score
AWS ECS on Fargate for Next.js API	Container-based hosting; fully managed orchestration with no servers to manage. Deployed behind an ALB for global, secure access	- High Uptime: Multi-AZ, automated failover. - Security: IAM roles, private subnets, easy compliance docs. - **Integration**: Works well with AWS CodePipeline, RDS, VPC.	3 + 3 + 2 = 8
AWS Lambda (Serverless) (optional)	Alternative for serverless Next.js – auto-scale, pay-per-use. Good for microservices or certain SSR endpoints.	Event-driven scale with no server management. - Cost-effective at sporadic loads. - Security: Minimum OS patching, Per function IAM	3 + 2 + 2 = 7

Cloud Services Vendor Selection Criteria

Selection Criteria Name	Selection Criteria Description	Selection Criteria Value (Weighting in Points)
Security & Compliance	Must meet PCI DSS, SOC 2, provide advanced encryption,	3

	auditing, identity & access controls	
Integration & Ecosystem	Vendor solutions must integrate with Next.js, RDS, VPC, IAM, WAF	3
Scalability	Ability to handle concurrency bursts with minimal manual intervention	3
Cost Transparency	Clear pricing model for compute, data transfer, storage	2

Cloud Services Recommended Vendors

Vendor Name	Vendor Strengths	Vendor Weaknesses	Products/Services Provided to Project	Aggregate Selection Criteria Score
Amazon Web Services	Industry-leading compliance - Rich ecosystem (Lambda, ECS, RDS, S3, CloudFront) - Flexible cost models, pay-as-you-go	Complexity in cost management - Proprietary environment	ECS Fargate for containers - AWS Lambda (optionally) - RDS for database - CloudFront, WAF, Route 53 for global distribution & DNS	3
Microsoft Azure	- Comprehensive compliance portfolio with industry certifications - Rich, integrated ecosystem (AKS, Azure Functions, SQL Database, CDN, Front Door) - Flexible cost models and enterprise agreements	- Complexity in managing hybrid deployments - Potential vendor lock-in and cost management challenges	- Azure Kubernetes Service (AKS) for container orchestration (similar to ECS Fargate) - Azure Functions as a serverless alternative to AWS Lambda - Azure SQL Database for managed database needs - Azure CDN, Azure Front Door, Azure WAF, and Azure DNS for global distribution and security	2

Cloud Services Deployment Challenges

Deployment Challenge	Deployment Challenge Description
Multi-Region Coordination	Setting up identical infrastructure across regions for best global coverage; can be more complex in code and cost.

Vendor Lock-In	Relying heavily on AWS-specific features (e.g., ECS, Lambda) can make future migrations harder.
Monitoring & Cost Management	Ensuring CloudWatch alerts, usage dashboards, and budgeting are in place to avoid unexpected bills
Improvement #2	Add AWS budgets along with Cloudwatch dashboards to monitor resource usage and alert cost thresholds (This recommendation comes from research findings that cost overruns are a significant risk in cloud projects; real-time monitoring tools help mitigate this by offering visibility and control.)

Technology Adoption Methods

Method Name	Method Description
Phased Migration	Gradually deploy services (e.g., dev -> staging -> prod) to validate performance and cost at each step
Well-Architected Reviews	Use AWS Well-Architected Framework to review reliability, security, cost optimization, performance, and operational excellence
Improvement #3	Add structured onboarding with sandbox environments and training for developers to ensure all developers are working at the same level of technical knowledge (Research indicated that a steep learning curve could delay the project. Structured onboarding improves productivity and ensures smoother adoption of CI/CD, IaC, and SSR patterns.)

Cost/Benefit Considerations

Benefits	Costs	Considerations
Automatic scaling & patching	Potentially higher cost at scale	Evaluate usage patterns carefully to rightsize infrastructure

High reliability (SLA 99.99%)	Data transfer (egress) fees may increase with global usage	Use monitoring and budgeting tools to control spending
Global low latency with CDN	Additional cost for advanced security/compliance features (e.g., WAF)	Plan for multi-region and multi-AZ deployments to ensure redundancy and performance
Mature security & compliance features (PCI DSS, SOC 2, etc.)	Skilled personnel/training needed for specialized cloud configurations	Factor in the overhead of regular compliance audits and certifications

University of Phoenix®

Supporting Research Report

Fill out each section with information relevant to your project. Be sure to include the name and purpose of your project.

Supporting Research Report for: Kenneth Quiggins

Project name: Kentech Banking

Purpose of project:

To design and deploy a scalable, secure, and high performing financial web platform using cloud-native technologies to support modern banking needs such as secure web portals, real-time transactions, and maintain compliance with financial industry regulations.

- **Executive Summary**

  The Kentech banking project aims to modernize the financial banking industry through a secure, responsive, and scalable web platform hosted in the cloud. The project adopts Amazon Web Services (AWS) for infrastructure, leverages Next.js for the frontend/backend framework, and utilizes Amazon RDS for secure, ACID-compliant relational data storage. The platform is designed to meet industry standards in security, performance, and global availability. The proposed solution replaces legacy systems with a modular, cloud-native architecture that simplifies maintenance and supports future growth.

- **Industry Background**

  The banking industry has seen a dramatic shift toward digital experiences over the past decade, a trend accelerated by evolving consumer expectations and competitive pressures.

  Financial institutions face increasing pressure to provide digital services with strong security and low latency. Traditional on-premises infrastructure is often too rigid and costly to scale, making cloud migration a necessary evolution. Regulatory compliance (e.g., PCI DSS, SOC 2) adds further complexity, as systems must be auditable, encrypted, and resilient. Cloud-native technologies now dominate the landscape, with providers like AWS enabling rapid deployment, security automation, and elasticity needed for modern financial operations.

  Recent data from Statista showed there has been a continued decline in people attending U.S. bank branches in person. In the fourth quarter of 2024, 45 percent of U.S. bank account holders reported conducting activities in person at a branch, a decrease from 53 percent from the first half of 2019 (Newsweek, 2025).

- **Technology Trends**

The financial services industry is undergoing a technological shift that influences everything from consumer expectations to operational strategy.

Influenced by rapid digital transformations during the pandemic, banks of all sizes are migrating to cloud services. This shift helps them meet customer demands, fend off competition, enhance efficiency, and accelerate business growth (PwC, 2025).

Between 2011 and 2021, the global percentage of adults with bank accounts rose from 51% to 76%, driven by digital and cloud technologies. This transformation reduced service costs significantly, exemplified by Nubank in Brazil, which serves over 100 million customers at an average operating cost of less than $2/month (Forbes, 2025).

Larger banks, through national and global consumer brands, are well-positioned to dominate profitable sectors like consumer banking and wealth management. However, specialized local banks still play vital roles by fostering community relationships (Forbes, 2025).

- ## Project Approach

This project will follow a modern agile methodology, ensuring iterative development and early user feedback throughout the process.

The platform will be developed in sprints using agile methodology. AWS will host the infrastructure with a multi-tier VPC network, RDS for relational data, and Fargate or Lambda for application logic. The frontend/backend will be built in Next.js to support SSR and SSG, ensuring global performance and security. Deployment pipelines (CI/CD) will be integrated via GitHub Actions or AWS CodePipeline. IaC will define all resources, ensuring consistency across environments. Security will follow best practices with IAM roles, security groups, encryption, and routine audits.

- ## Alternative Approach

An alternative to this solution could involve using a Platform-as-a-Service (PaaS) like Heroku or Firebase for hosting, and a non-relational database like Firestore or MongoDB Atlas. While easier to manage initially, this approach presents limitations in ACID compliance, data structure flexibility, and vendor lock-in. Additionally, compliance with financial standards may be harder to prove without control over network and storage configurations, making it less favorable for banking applications.

.

- ## Impact Analysis

Implementing this cloud-native financial platform is expected to drive both technical and business-level benefits.

Positive Impacts:

- Enhanced user experience due to faster page loads and responsive design.
- Improved security and compliance through encryption and role-based access control.

  - Reduced downtime via automated failover and multi-AZ deployments.
  - Faster deployment cycles due to IaC and CI/CD.
- **Negative Impacts:**
  - Higher learning curve for teams unfamiliar with AWS and Next.js.
  - Potential for increased cost if resource usage is not optimized.
- ## Risk Analysis

  High Risks:

  - **Compliance Failure:** Incomplete audit trail or data exposure could violate regulations. Mitigation: Use AWS Config, CloudTrail, and KMS.
  - **Cost Overrun:** Misconfigured services may lead to unexpected billing. Mitigation: Budgets, alerts, and usage reviews.
  - **Performance Bottlenecks:** Improper configuration of scaling policies or DB reads. Mitigation: Load testing and performance tuning in test phases.

  Medium Risks:

  - **Developer Ramp-up:** Team may need training on AWS services and SSR/SSG patterns.
  - **Vendor Lock-in:** Reliance on AWS-native services makes future migration more complex.

  Low Risks:

  - **Delays in Agile Sprints:** May occur if the feature scope is not well-defined. Mitigation: Tight backlog grooming and stakeholder reviews.

References:

Amazon EC2 Autoscaling. (2024): https://docs.aws.amazon.com/autoscaling/ec2/userguide/create-asg-launch-template.html

Chat-GPT. (n.d.): https://chatgpt.com

Cloud Maintenace 101. (2024): https://www.comptia.org/content/articles/cloud-maintenance-101-checking-the-pulse-of-your-cloud-technology

Continuous integration. (n.d.): The Free Encyclopedia: http://en.wikipedia.org/wiki/Continuous_integration

Simplify your way to growth through a cloud-powered bank. (2025): https://www.pwc.com/us/en/industries/financial-services/library/cloud-banking-trends.html

Top 10 Trends For Banking In 2025 – The Future Is Back. (2025): https://www.forbes.com/sites/michaelabbott/2025/01/13/top-10-trends-for-banking-in-2025--the-future-is-back/