SOLUTION ARCHITECTURE AND DESIGN FOR the chisira PORTFOLIO WEBSITE.

Template Revision History

| Ver. | Date | Author | Comments |
| --- | --- | --- | --- |
| 1.0 | 1-April-2023 | XXXX | XXXX |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

DESIGN AND ARCHITECTURE Document For CHISIRA PORTFOLIO WEBSITE

|  |  |
| --- | --- |
| Prepared by: | Levis Shitubi Chisira |
| Prepared for: |  |
| Date submitted: |  |
| Business Sponsor: |  |
| Project Manager: |  |
| Solution Architect: |  |
| Filename: |  |
| Document Number: |  |
|  |  |

Table of Contents

[1 APPROVALS & REVISION HISTORY 5](#_Toc378163392)

[1.1 Document Approval 5](#_Toc378163393)

[1.2 Revision History 5](#_Toc378163394)

[2 PURPOSE 6](#_Toc378163395)

[3 REFERENCED DOCUMENTS 6](#_Toc378163396)

[4 GLOSSARY 6](#_Toc378163397)

[5 SOLUTION OVERVIEW 7](#_Toc378163398)

[5.1 Solution Domain 7](#_Toc378163399)

[5.2 Architecture Overview 7](#_Toc378163400)

[5.3 Scope 9](#_Toc378163401)

[5.3.1 In-Scope 9](#_Toc378163402)

[5.3.2 Out-Scope 9](#_Toc378163403)

[5.4 Integration Interfaces 9](#_Toc378163404)

[5.5 Future Architecture 9](#_Toc378163405)

[5.6 Transition Planning 9](#_Toc378163406)

[6 ARCHITECTURE GOALS & CONSTRAINTS 10](#_Toc378163407)

[6.1 Safaricom Enterprise Architecture 10](#_Toc378163408)

[6.2 Architectural Assumptions and Decisions 10](#_Toc378163409)

[6.2.1 Assumptions & Decisions 10](#_Toc378163410)

[6.2.2 Constraints 10](#_Toc378163411)

[6.2.3 Open Issues 10](#_Toc378163412)

[6.3 Solution Architecture Attributes 10](#_Toc378163413)

[6.3.1 Technology 10](#_Toc378163414)

[6.3.2 Patterns 11](#_Toc378163415)

[6.3.3 Common Services 11](#_Toc378163416)

[6.3.4 Common Components 11](#_Toc378163417)

[6.3.5 Portability 11](#_Toc378163418)

[6.3.6 Capacity 11](#_Toc378163419)

[6.3.7 Performance 11](#_Toc378163420)

[6.3.8 Availability and Reliability 11](#_Toc378163421)

[6.3.9 Scalability 11](#_Toc378163422)

[6.3.10 System Management, Monitoring and Administration 11](#_Toc378163423)

[6.3.11 BC & DR 11](#_Toc378163424)

[6.3.12 Other Solution Architecture Issues 12](#_Toc378163425)

[7 APPLICATION ARCHITECTURE 13](#_Toc378163426)

[7.1 Application Layers 13](#_Toc378163427)

[7.2 Logical Architecture Model 13](#_Toc378163428)

[7.3 Physical Architecture Model 14](#_Toc378163429)

[7.4 Common Service Specifications (Optional) 14](#_Toc378163430)

[7.5 Component Models 15](#_Toc378163431)

[7.5.1 Component Diagrams 15](#_Toc378163432)

[7.5.2 Sequence Diagrams 15](#_Toc378163433)

[7.6 Walk-Through Models 15](#_Toc378163434)

[8 DATA ARCHITECTURE 16](#_Toc378163435)

[8.1 Data Flow and Context Diagrams (Optional) 16](#_Toc378163436)

[8.2 Conceptual/Logical Data Model 16](#_Toc378163437)

[8.3 Authoritative Data Sources 16](#_Toc378163438)

[8.4 Physical Data Elements 16](#_Toc378163439)

[8.5 XML Resources Guidance 16](#_Toc378163440)

[8.6 Data Migration Guidance (Optional) 16](#_Toc378163441)

[9. SECURITY ARCHITECTURE 17](#_Toc378163442)

[9.1 Security Solution Overview 17](#_Toc378163443)

[9.2 Security Architecture Goals and Constraints 17](#_Toc378163444)

[10. INFRASTRUCTURE 18](#_Toc378163445)

[10.1 Deployment Models 18](#_Toc378163446)

[11 APPENDIX 19](#_Toc378163447)

[11.1 Appendix A: Table Structures and Stored Procedures 19](#_Toc378163448)

[11.2 Appendix B: Web Services Artifacts 19](#_Toc378163449)

[11.3 Appendix C: Server Resource Information 19](#_Toc378163450)

[11.4 Appendix D: Other Required Artifacts 19](#_Toc378163451)

# 1 APPROVALS & REVISION HISTORY

This document has been approved as the official solution architecture document for the Chisira Portfolio Website project, and accurately reflects the current understanding of solution design and architecture.

## 1.1 Document Approval

| Ver. | Name | Role/Department | Signature | Date |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## 1.2 Revision History

| Ver. | Date | Author | Comments |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 2 PURPOSE

The purpose of this document is to define a high-level solution design and architecture for the Bennavi Softsolution Portfolio Website*.*

# 3 REFERENCED DOCUMENTS

<Insert the name, version number, description, and physical location of any documents referenced in this document. Add rows to the table as necessary.>

|  |  |  |
| --- | --- | --- |
| **Document Name** | **Description** | **Location** |
| *<Name & Version>* | *<Document description>* | *<Document URL>* |
|  |  |  |
|  |  |  |

*Table 1: Referenced Documents*

# 4 GLOSSARY

<Insert the definitions of terms and acronyms used in this document. Add rows to the table as necessary.>

|  |  |
| --- | --- |
| **Term** | **Definition** |
| *<Insert Term>* | *<Provide definition of term/acronyms used in this document.>* |
|  |  |
|  |  |

*Table 2: Glossary/Key Terms*

# 5 SOLUTION OVERVIEW

## 5.1 Solution Domain

*The area of focus in this project is to design and develop a web based application that will work as my Tech business portfolio. Users and customers will be able to access my information over the system ,history of the business,subscribe to my newsletters ,and access information about the services I offer – WiFi installations ,and software development.*

*Customers would also be able to pay for services over the portfolio and check the physical address too.*

## 5.2 Architecture Overview

*<Provide one or more diagrams depicting an overview of the solution architecture with the necessary descriptive text. The diagram(s) should depict the major components of the solution and the relationships between the components, input and output data flows, major processes, functions, and system tasks. Identify major Safaricom enterprise systems, infrastructure, and platform technology components.>*



*Figure 1: Architecture Overview Example 1*

## 5.3 Scope

### 5.3.1 In-Scope

*The system is to only serve the customers and users withing Kenya ,and will only give information on which services I offer – with information on my location ,address ,about me ,payments ,and even screenshots of the sample products I have done before.*

### 5.3.2 Out-Scope

*The Portfolio will not be able to allow a user or a customer order a product remotely. It is only to give information on the products ,and how to contact me on any inquiries.*

## 5.4 Integration Interfaces

*The following are some of the integration interfaces between the solutions system that be the communication channel – that enables the exchange of data and functionality within the system.*

|  |  |  |
| --- | --- | --- |
| ***Target System*** | ***Interface Type*** | ***Break*** |
| *Daraja API*  *File transfer.*  *Database interface*  *Contact interface*  *google maps* | *Upstream*  *Downstream*  *Upstream*  *upstream*  *downstream* | *NO*  *Yes*  *No*  *No*  *Yes* |

*Table 3: High Level Interfaces*

## 5.5 Future Architecture

*The potential new feature or component in the near future is the functionality on the ordering products or services over the web based application portfolio.*

## 5.6 Transition Planning

*Currently , customer and product information is recorded manually – where a customer has to access my physical offices to inquire where about my services so as to make an order. With the new system I will take the following measures to get a smooth transition :*

1. *Planning – Assessment is done to get the goals and objectives of transiting to the new system.*
2. *Designing and Developing – involves designing and developing the new system on target that meet the required standards.*
3. *Training and communication – all the stakeholders are trained and given support to make sure they can use the new system. Also communicating to the stakeholders about the transition , and the benefits of the new system.*
4. *Implementation – deploying or rolling out the new system to the users and other stakeholders to use.*
5. *Evaluation – motoring the performance and effectiveness of the new system*

# 6 ARCHITECTURE GOALS & CONSTRAINTS

Description of the goals and constraints of the Solution Design explained further here below.

## 6.1 Safaricom Enterprise Architecture

*The following layers are part of the solution Design of this system in relation to Safaricom Enterprise Architecture:*

1. Business architecture – this entity is focused on serving the customers on Tech based services. WiFi installation ,system development and support for companies,and technology trainings are some of the services offered.
2. Application architecture – this web-based application system is on target to support the business processes and transactions. This include inquiries from the customer , payment made both on the system and off the system ,products,and location.
3. Data architecture – this layer is responsive for the data structures used to store and manage data. This includes data related to customer accounts , product , billing or payment ,and other stakeholders like admin.
4. Technology architecture – involves the technology infrastructure needed to give support to the other architectures and layers. This includes the hardware , software ,networking ,and security components

## 6.2 Architectural Assumptions and Decisions

### 6.2.1 Assumptions & Decisions

*The following are the assumption and decision aspects of this project:*

***Assumptions.***

1. The users and the customers have access to smartphones or computers systems.
2. Users and customers are able to read and understand english.
3. Contented customers make referrals to others who don’t know about the services.
4. Project design and development would be finished within three months.

***Decisions:***

1. *Design and develop a web based application system.*
2. *Use English as the main language.*
3. *Design and come up with very efficient UX/UI.*
4. *Allocate a short time period to the new system transition.*

### 6.2.2 Constraints

*The limitation to smart phones and computer access among the users and customer limits this architecture to serving those users. Yes sms messages functions may be used but still some of the components in the system may be broken example google maps for location.*

### 6.2.3 Open Issues

*The following are issues on this solution architecture :*

1. Government Policies : financial transactions information shall not be hosted or managed from outside the country. This policy may prompt to either decide on an architecture that will allow fetching data from more than one location or have a system that is locally hosted and managed.
2. Computer literacy : some of the customers are computer illiterate which may make the system to be less effective to them

## 6.3 Solution Architecture Attributes

The following are the properties that the the quality and effectiveness of the solution architecture is defined:

1. Security: the solution architecture of the project considers control on who can access the details on the services of offered – through authentications on the system. Users with no account are allowed to sign up.
2. Interoperability : Being a web based application , the system can run on different web browsers and operation system. The responsiveness technique is used to ensure accommodation of different gadgets.
3. Performance : the solution architecture is determined on effective performance – support team information and addresses on system allows user to ask any inquiries if there is any.
4. Scalability : the solution architecture is open to improvement and building a new version of it.

### 6.3.1 Technology

|  |  |  |  |
| --- | --- | --- | --- |
| ***Vendor*** | ***Product*** | ***Version*** | ***Status*** |
| Hp  #  #  Safaricom  #  # | Elite Book 2570p  Linux  MYSQL  router  PHP  Bootstrap  HTML, java script  Reactjs | Core i5  Mint  latest  4G Pocket Mifi  5  latest | good  latest  updated  latest  good |

*Table 4: Required Technologies*

### 6.3.2 Patterns

***Layered Architecture Pattern*** *: different layers like data access layer , Business logic layer ,and presentation layer e.t.c are separated - inter-linking them to make up a system ,making it easy to modify and test individual layers.*

### 6.3.3 Common Services

*The following are the common services ,functions or components in this solution architecture:*

1. Messaging and Event Services: These services enable communication between different components of the system by sending and receiving messages or events.
2. Logging and Monitoring Services: These services provide a way to log system events and monitor system performance, errors, and issues.
3. Data Storage Services: These services provide a way to store and retrieve data from different types of data stores such as relational databases, No-SQL databases, and data warehouses.
4. Caching Services: These services provide a way to cache frequently accessed data or results to improve system performance and reduce response times.
5. Content Delivery Services: These services provide a way to deliver content such as web pages, images, and videos to users around the world with low latency and high availability.
6. Authentication and Authorization Services: These services provide a secure way to authenticate and authorize users to access the system or application.

### 6.3.4 Common Components

The common components to be used are as follows:

1. User Interface (UI) Components: These components provide a way to create and display user interfaces such as buttons, forms, tables, and menus. Examples of these components include Reactjs, Angular, and Vue.
2. Data Access Components: These components provide a way to interact with databases and data stores by performing CRUD (Create, Read, Update, and Delete) operations.
3. Messaging Components: These components provide a way to send and receive messages or events between different components of the system. Examples of these components include JMS, AMQP, and MQTT.
4. Security Components: These components provide a way to secure the system by encrypting data, validating user inputs, and preventing unauthorized access. Examples of these components include Spring Security, Apache Shiro, and Oauth.
5. Logging and Monitoring Components: These components provide a way to log system events and monitor system performance, errors, and issues. Examples of these components include Log4j, Logback, and Prometheus.
6. Integration Components: These components provide a way to integrate different systems or applications by exchanging data and messages between them. Will use Apache Camel.

### 6.3.5 Portability

*Portability refers to the ability of the software to run on a different platforms or operating systems with minimal or no changes. The following are ways will use to achieve portability:*

1. Cross-Platform Technologies: use cross-platform technologies like Reactjs or Nodejs that can run on multiple operating systems.
2. Dependencies: avoid platform-specific dependencies such as libraries or frameworks that are only available on certain platforms.
3. Platform-Independent APIs: use platform-independent APIs (Application Programming Interfaces) that can abstract away the differences between different operating systems.
4. Different Platforms Testing:perform test on different platforms and operating systems to identify any issues or incompatibilities.

### 6.3.6 Capacity

*The projected capacity of this solution architecture is around 100000 users every month. The following are the measures taken to address capacity in the software:*

1. Scalability: the system can handle increasing workload or user traffic by adding additional resources such as servers or instances.
2. Performance Optimization: This can involve optimizing code, reducing latency, and using caching or load balancing techniques.
3. Capacity Planning: estimating the resource requirements of a system based on expected workload or user traffic.

### 6.3.7 Performance

*With the following ways of addressing performance the solution architecture will up to task on performance .*

1. *Performance Testing: measuring the system's performance under different conditions to identify bottlenecks and potential issues – to identify and solve the issue of performance early.*
2. *Code Optimization: reducing execution time, improving memory utilization, and minimizing I/O operations. This involves optimizing algorithms, using caching, and avoiding unnecessary processing.*
3. *Resource Optimization: This can involve reducing unnecessary processes or services, optimizing resource allocation, and implementing resource pooling.*
4. *Scalability.*

### 6.3.8 Availability and Reliability

1. Redundancy:this involve using redundant servers, data centers, or network connections to ensure that the system remains available even if one component fails.
2. Disaster Recovery: Involves backing up data regularly, implementing redundant systems, and creating contingency plans for restoring the system in case of a major failure.
3. Fail-over: Load balancers, clustering, or other mechanisms are used to detect failures and automatically switch to redundant components.
4. Monitoring and Alerting: This can involve using tools such as Nagios, Zabbix, or New Relic to monitor the systems performance.

### 6.3.9 Scalability

*Here are some ways this solution design accommodates growth :*

1. Database Sharding: The database Is partitioned across multiple servers to handle increasing data volumes or traffic.
2. Vertical Scalability: This involves involve the use of cloud-based services such as AWS or Azure to provision additional resources on-demand.
3. Horizontal Scalability: This involves load balancing to distribute traffic across multiple instances or servers, or using containerization technologies such as Docker or Kubernetes to deploy and manage instances or servers.

### 6.3.10 System Management, Monitoring and Administration

*This process maintains ,monitor ,and manages a software system. Here are some ways this solutions architecture will address the system management, monitoring ,and administration:*

1. *Configuration Management: managing the configuration of the system, including software versions, updates, and patches – with tools like Ansible, Puppet, or Chef .*
2. *Backup and Recovery: involves using tools such as AWS Backup, Azure Backup, or Veeam Backup to manage backups and restore data.*
3. *Performance Monitoring: Involves monitoring the system's performance, including CPU, memory, disk usage, and network activity – with tools like Nagios.*
4. *Security Management: the system manages the access control, authentication, and authorization – with tools like Keycloak.*

### 6.3.11 BC & DR

*Business continuity and disaster recovery is the process and procedure designed to ensure the availability and reliability of a software system in the face of unexpected events. With this solution architecture the following are ways how BC & DR are addressed :*

1. Risk Assessment: This involves conducting a risk assessment of the system, identifying potential threats, and implementing measures to mitigate those risks.
2. Planning Business Continuity: Involves developing plans and procedures to ensure that the system remains available and operational in case of a disruption – using measures like redundant systems, backup and recovery procedures, and disaster recovery planning.
3. Disaster Recovery Planning: involve implementing measures such as offsite backups, redundant systems, and fail-over procedures.

### 6.3.12 Other Solution Architecture Issues

*The following are the other Solution architecture issues :*

1. *Compliance: involves ensuring that the system complies with the government legal framework, and industry standards – example data protection Act. This can involve implementing measures such as auditing, monitoring, and reporting.*
2. *Data Security: protecting sensitive data from unauthorized access, use, disclosure, or destruction by implementing access controls, encryption, and data loss prevention.*
3. *User Experience(UX): involves designing the software system to meet the needs and expectations of the users. How the user feels about the product.*
4. *Cost: This involves implementing measures such as cost estimation, resource management, and optimization – to manage the cost of the software.*

# 7 APPLICATION ARCHITECTURE

This is a description of the solution’s application architecture (major solution components and their relationships). Include significant diagrams and specifications.

## 7.1 Application Layers

*<Organize the system components involved according to application layers>*

## 7.2 Logical Architecture Model

*<Create a logical architecture model illustrating data exchange and communication protocols and standards between components and systems in the design>*



*Figure 3: Sample Logical Architecture Model Diagram*

Graphical user interface, diagram

Description automatically generated

*Figure 3: Sample Logical Architecture Model Diagram 2*

## 7.3 Physical Architecture Model

*<Create a physical architecture model illustrating the physical boxes and components involved in the system including DB servers, application servers, web servers, HA & Load Balancer boxes etc>*



*Figure 4: Sample Physical Architecture Model Diagram*

## 7.4 Common Service Specifications (Optional)

*<If the solution integrates common services, describe how all common services identified by the solution architecture’s common service attributes are integrated into the solution’s application architecture>*

## 7.5 Component Models

### 7.5.1 Component Diagrams

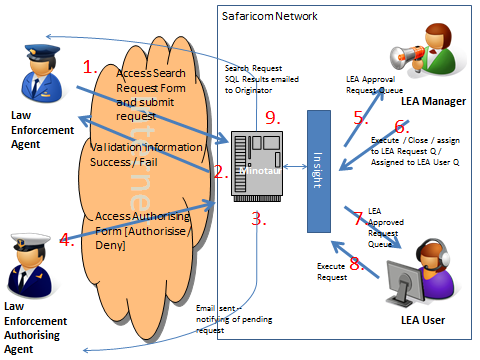
*<Provide component models illustrating static views of component relationships and define all objects and relationships depicted by the models.>*

### 7.5.2 Sequence Diagrams

*<Provide sequence models illustrating dynamic views of component interaction. Include models that illustrate interaction of components across application layers supporting the significant and central use case scenarios.>*

## 7.6 Walk-Through Models

*<Provide walkthrough models to trace system execution and validate the solution’s application architecture and component interaction required to implement significant and central use cases and business processes (illustrated by a numbered sequence of defined events) defined by the solution requirements.*



*Figure 5: Sample Workflow Diagram.*

# 8 DATA ARCHITECTURE

## 8.1 Data Flow and Context Diagrams (Optional)

*<Provide context and data flow diagrams showing data flows between a generalized application within the domain and the other entities and abstractions with which it communicates.>*

## 8.2 Conceptual/Logical Data Model

*<Provide conceptual (describing the logical grouping of the basic data building blocks of the solution) and logical (describing the major processes and data requirements of the business) data models.>*

## 8.3 Authoritative Data Sources

*<Identify the authoritative data sources required for access during this project.>*

## 8.4 Physical Data Elements

*<Provide guidelines for extending the logical model with a physical data model to define the physical representation of the data.>*

## 8.5 API Resources Guidance

*<Provide the schemas and guidelines for developers to follow when designing and creating XML/JSON resources – includes web services resources like wsdl, JSON and xsd files>*

## 8.6 Data Migration Guidance (Optional)

*<If the solution involves data migration, define the appropriate transition plan to indicate data migration sequencing requirements in relationship to the solution’s transition from current baseline architecture to the target architecture.>*

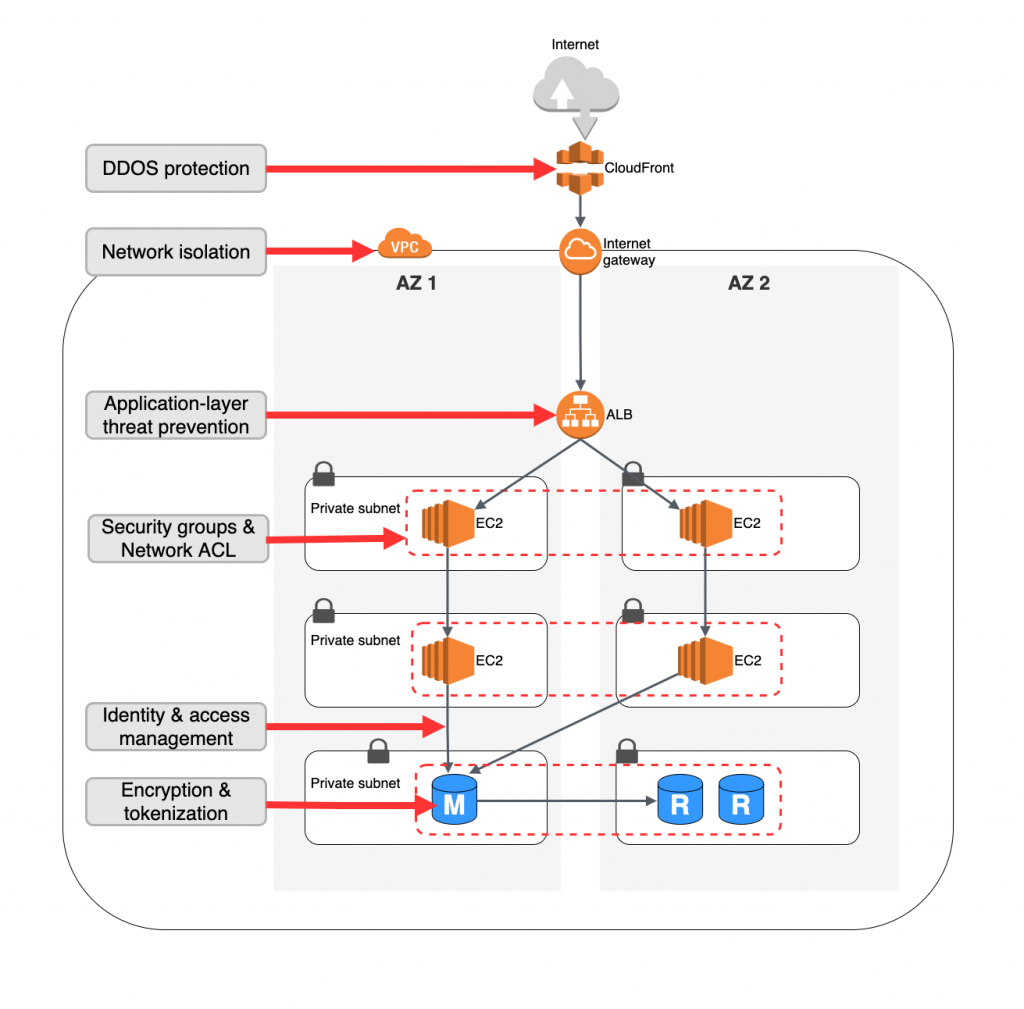
# 9. SECURITY ARCHITECTURE

## 9.1 Security Solution Overview

*<Provide a high-level solution overview and description of the security architecture. Identify and describe how the security architecture meets the solution’s security requirements.>*

## 9.2 Security Architecture Goals and Constraints

*<Identify and describe the significant and central security goals and constraints of the solution’s security architecture.>*



*Figure: Sample Logical Architecture Model Diagram*

# 10. INFRASTRUCTURE

*<Map application architecture deployment models to hardware and software infrastructure specifications including memory and CPU specifications required to meet volume and performance requirements, including architecture guidance and specifications for:*

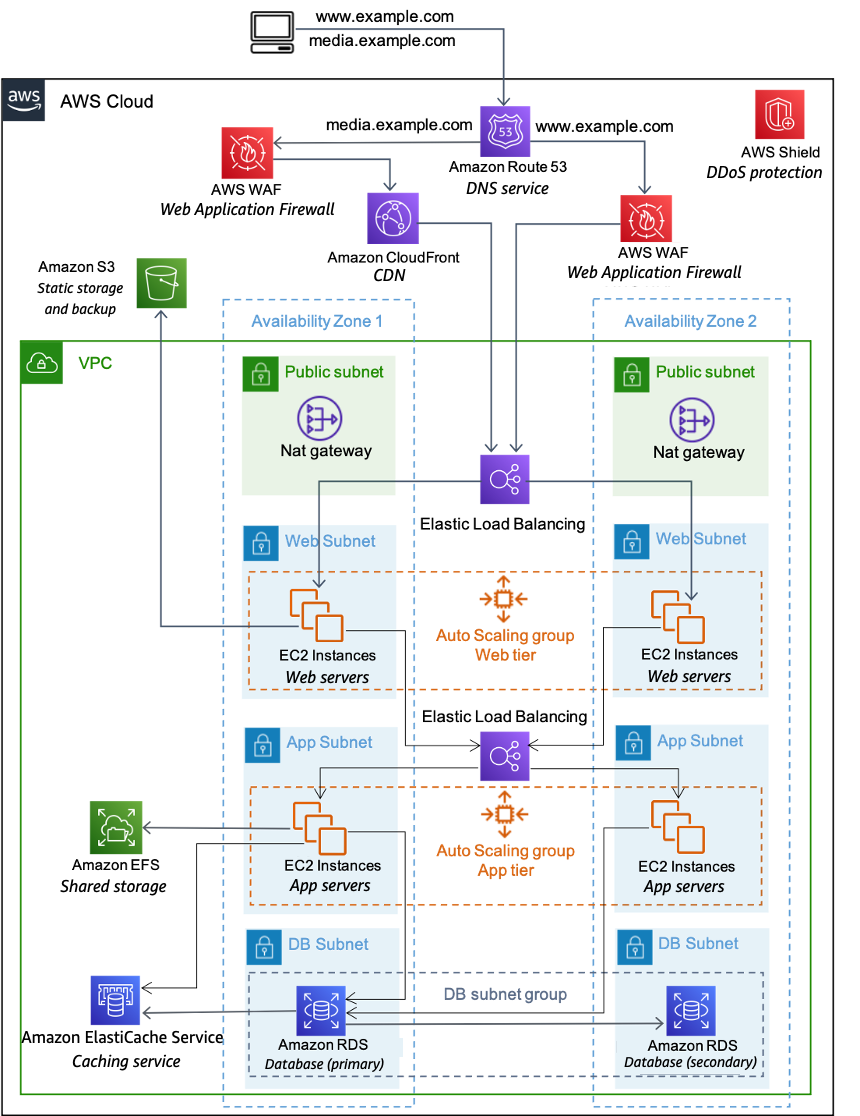
1. *Software*
2. *Hardware*
3. *Network*
4. *Middleware*

*Provide infrastructure architecture guidance and specifications for all environments required for developing, testing, deploying, and operating the solution.>*

## 10.1 Deployment Models

*<Describe how the application architecture is deployed into one or more physical network (hardware) configurations.*

*Include one or more diagrams to illustrate significant and central components of the infrastructure architecture, these diagrams should be easily cross-referenced with the infrastructure architecture overview diagram.*



# 11 APPENDICES

## 11.1 Appendix A: Table Structures and Stored Procedures

<Insert structures of the key database tables, views and queries used in the solution. Insert stored procedures and triggers, if they are to be used.>

## 11.2 Appendix B: Web Services Artifacts

<Insert key web services artifacts used in the solution, for example wsdl and xsd interfaces>

## 11.3 Appendix C: Server Resource Information

<Include detailed technical information about the specific server boxes to be used.>

## 11.4 Appendix D: Other Required Artifacts

<Insert any other required solution artifacts.>