THE GENESIS!

"The power to new beginnings"

Salutations Genesis members the document outlines the individual of the members for the remaining phases.

Phase 3: Scenarios, and User Stories • Identify target users using personas (fictional user profiles). • Write scenarios demonstrating real-world user interactions. • Develop user stories (structured as "As a [user], I want to [task] so that [goal]"). • Prioritize features based on user impact. Page 3 of 8 • Map features to functional and non-functional requirements. **Deliverables** • User Personas Document • User Scenarios & Story Mapping	**Phase 4:** Software Architecture & System Design, Software Architecture) Milestones • Define the software architecture (Monolithic, Microservices, Layered, etc.). • Create system decomposition diagrams (module interactions). • Choose the distribution architecture (client-server, P2P, cloud-hosted). • Document technology stack selection (programming language, database, frameworks). • Consider scalability, performance, and maintainability. **Deliverables** • System Architecture Diagram (UML, ERD, or Flowchart) • Decomposition & Distribution Architecture Document & technology stack justification

TO TAKE NOTE:

- **Sections to include**:
  - Name & Background
  - Goals & Pain Points
  - Preferred Technologies
  - Usage Scenarios

User Scenarios & Story Mapping

- **Sections to include**:
  - Narrative flow of user experience
  - Key interactions & system responses

Our Project Tester will present 3 persona and 3 scenarios to link with the current project.

E.g. The 3 personas and scenarios can be with the layout design in terms of the simplicity of it makes the user be able to use it easier, the interactive of the user and software.

Example:

Persona: Lerato Mokoena – The Student-Athlete

- **Profile**: Lerato is a 20-year-old third-year Biochemistry student at WSU and the captain of the university's netball team.
- **Challenges**:
  - Struggles to balance academic responsibilities with athletic commitments.
  - Finds it difficult to create an effective study timetable that accommodates her training schedule.
  - Experiences fatigue due to lack of structured rest periods.
- **Needs**:
  - A virtual assistant that provides personalized tutoring and helps create an optimized timetable.
  - A system that considers class schedules, training sessions, and personal commitments.

- Tools to monitor academic and athletic milestones.

User Story

As Lerato, I want a virtual assistant that not only offers personalized tutoring but also helps me create a balanced schedule. This assistant should consider my academic and athletic commitments, ensuring I can excel in both areas without compromising my well-being.

Scenario: Lerato's Personalized Academic and Athletic Scheduler

- **Problem**: Lerato often misses assignment deadlines and feels unprepared for matches due to poor time management.
- **Solution Approach**:
  - Implement a digital platform that integrates academic and athletic schedules.
  - Provide access to subject-specific tutors familiar with student-athlete challenges and include features that prompt rest and recovery periods to prevent burnout.
- **Objective**: Enhance Lerato's academic performance while ensuring she remains at peak athletic condition.
- **Steps to Achieve Objective**:
  1. Lerato logs into the UniConnect platform.
  2. Inputs her class timetable, training sessions, and other commitments.
  3. The platform generates a balanced schedule, allocating time for study, practice, rest, and tutoring sessions.
  4. Lerato receives reminders and adjusts her schedule as needed, maintaining flexibility.

TO TAKE NOTE by our System designer

Feature Prioritization & Requirements Document

- What is MoSCoW Prioritization? | Overview of the MoSCoW Method
- **Sections to include**:
  - Feature List (Must-Have, Should-Have, Could-Have)
  - Functional & Non-Functional Requirements

System Architecture Diagram (UML, ERD, or Flowchart)

- **Sections to include**:
  - System Components
  - Data Flow
  - Integration Points

System Architecture Diagram (UML, ERD, or Flowchart)

This section should visually and descriptively present how the system is structured and how various components interact. The diagrams provide a blueprint that guides development and ensures that stakeholders understand the system's design.

Expected Elements:

System Components

Identify and illustrate the major components or modules of the system.

Examples include User Interface, Database, APIs, Authentication Module, Reporting Engine, etc.

Clarify the role and responsibilities of each component.

Data Flow

Show how data moves through the system from input to output.

Use Data Flow Diagrams (DFDs) or UML Activity Diagrams to map out the flow.

Represent processes, data stores, and external entities clearly.

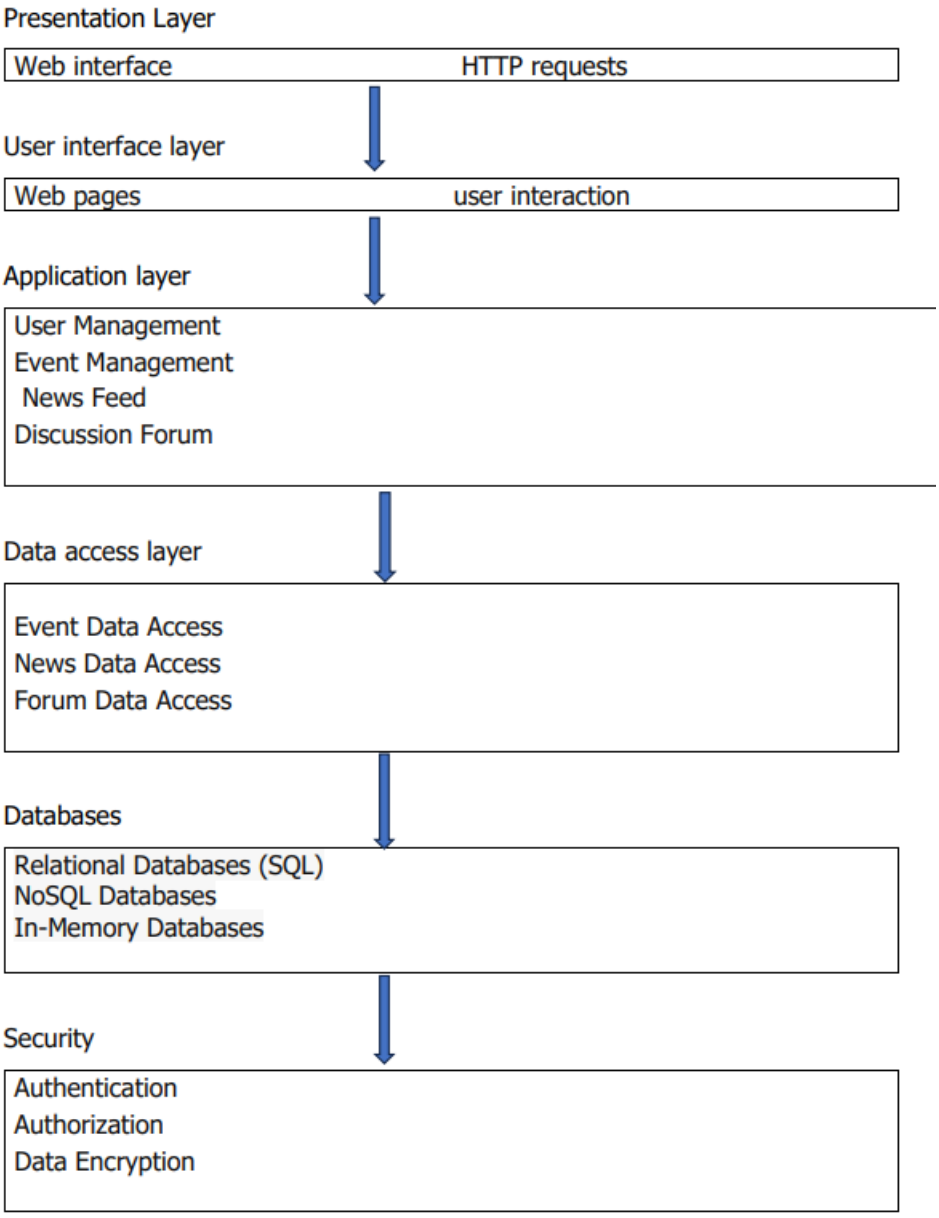
Integration Points

Indicate where and how the system integrates with external systems or services.

Examples: Payment gateways, third-party APIs, external databases, or user authentication providers.

Highlight communication protocols (e.g.API, FTP).

Software Architecture Model



TO TAKE NOTE by our Front-End Developer

To fully complete the **Decomposition & Distribution Architecture Document**

- **Sections to include**:
  - Service Layer Breakdown
  - Deployment Model
  - Communication Between Components

based on the image you've provided; you need to address the following

---

1. Service Layer Breakdown

This section describes how the system's services are logically grouped and structured. Steps to complete:

- **Identify Major Services:**
  - Define core services/modules such as Authentication, User Management, Data Processing, Notification Service, etc.
- **Break Down into Layers:**
  - Common layers include:
    - **Presentation Layer**
    - **Business Logic Layer**
    - **Data Access Layer**
    - **Integration Layer**
- **Document Responsibilities:**
  - Clearly describe the responsibilities of each layer/service.
  - Indicate which modules interact with others and how responsibilities are divided.
- **Use Diagrams:**
  - Optionally include a **layered diagram** to visualize the architecture and flow between layers.

---

2. Deployment Model

This part explains how the system will be physically or virtually deployed. Steps to complete:

- **Choose a Deployment Style:**
    - Examples: Monolithic, Client-Server, Microservices, Cloud-Native, Serverless, etc.
- **Define Deployment Environment:**
    - Specify environments (Development, Testing, Staging, Production).
    - Indicate whether deployment is on-premises, in the cloud (e.g., AWS, Azure), or hybrid.
- **List Components & Hosting Details:**
    - Identify where each component or service will be hosted.
    - Include servers, databases, containers (e.g., Docker), or serverless platforms.
- **Draw a Deployment Diagram:**
    - Use a **UML Deployment Diagram** or similar format.
    - Show nodes (servers, devices), software components deployed, and communication paths.
- **Include Security & Availability Considerations:**
    - Highlight firewalls, load balancers, backup servers, or disaster recovery plans.

---

3. Communication Between Components

This section outlines how various components interact and exchange data. Steps to complete:

- **Define Interactions:**
    - Describe what components need to communicate (e.g., front-end to API, API to database).
    - Clarify synchronous (e.g., HTTP) vs. asynchronous (e.g., message queues) communication.

- **Specify Protocols & Formats:**
    - Identify communication protocols
    - Define data formats.
- **Identify Interfaces:**
    - Document the APIs or interfaces exposed by each service/component.
    - Include any third-party integrations.
- **Include a Communication Diagram:**
    - Use a **Sequence Diagram**, **Component Diagram**, or **Interaction Diagram**.
    - Illustrate request/response flow and component dependencies.

TO TAKE NOTE by our Back-End Developer

To successfully complete the **Technology Stack Justification** section as outlined in the image, a **back-end developer** is expected to provide a well-reasoned, technically sound justification for each element of the stack used in the system.

Technology Stack Justification

- **What is a technology stack and why the right one matters - LogRocket Blog**
-
- **Sections to include**:
    - Chosen Programming Languages & Frameworks
    - Database Selection Rationale
    - Scalability & Performance Considerations

Detailed Structured Document can include

1. Chosen Programming Languages & Frameworks

Objective:

Explain which back-end programming language(s) and frameworks are chosen and why they are most suitable for the system.

Steps to complete:

State the Language and Framework:

E.g., Java with Spring Boot, Python with Django, Node.js with Express.js.

-Justify the Choice Based on:

-Performance & Scalability

- Ecosystem & Libraries

- Community Support

-Learning Curve & Team Expertise and security Features

---

2. Database Selection Rationale

Objective:

Provide a clear justification for the chosen database system (SQL or NoSQL), including its type and structure.

Steps to complete:

-Specify the Database Type:

E.g., MySQL, PostgreSQL, MongoDB, Firebase, etc.

- Explain the Reasoning:

-Data Structure Suitability (e.g., relational vs. document-based)

- Scalability and Storage Needs

-Read/Write Performance

-Consistency, Availability, Partition Tolerance (CAP theorem considerations)

-Ease of Integration with Chosen Stack

---

3. Scalability & Performance Considerations

Objective:

Show how the selected technologies support future growth, high performance, and system efficiency.

Steps to complete:

-Scalability Plans

How will the back end handle increasing users and data? (horizontal scaling, microservices, load balancing)

-Performance Optimization:

-Caching mechanisms (e.g., Redis)

- Asynchronous processing (e.g., message queues like RabbitMQ or Kafka)

- Efficient query handling (e.g., using indexes, ORM tuning)

-Deployment Strategies

-Use of containerization (e.g., Docker, Kubernetes)

-CI/CD pipelines for seamless updates and auto-scaling with cloud platforms

IF THERE IS ANY COMFUSION PLEASE FEEL FREE TO CONTACT THE PROJECT MANAGER
