Title: Online Bank Management System

- **Subtitle:** Ex. 2 Work Break-down Structure (Process Based, Product Based, Geographic Based and Role Based) and Estimations using FreePlane and python
- **Name:** Karan Sehgal
- **Registration No:** 22BCE3939
- **Team No: 24**
- **Course/Subject:** Software Engineering Lab (BCSE301P)
- **Instructor's Name:** Dr. Mehfooza M

Date of Submission: 06/01/25

Introduction

The Online Banking Management System (OBMS) is a comprehensive project aimed at streamlining banking operations and improving user experiences across multiple access levels—Admin, Manager, and Customer. To ensure the project is managed efficiently and delivered successfully, a structured approach using Work Breakdown Structures (WBS) has been employed. The WBS divides the project into manageable components, focusing on key aspects like processes, products, geography, and roles, making it easier to allocate resources, track progress, and ensure accountability.

This document outlines four types of WBS tailored for OBMS, each serving a unique purpose:

1. **Process-Based WBS:** Breaking down the project into sequential phases.
2. **Product-Based WBS:** Organizing tasks by deliverables and system components.
3. **Geographic-Based WBS:** Assigning tasks based on team location and regional deployment.
4. **Role-Based WBS:** Dividing responsibilities according to team roles.

These WBS structures provide a clear roadmap for the project, ensuring all stakeholders have a well-defined understanding of their roles and tasks, leading to a more efficient and successful project delivery.

1. Process-Based WBS

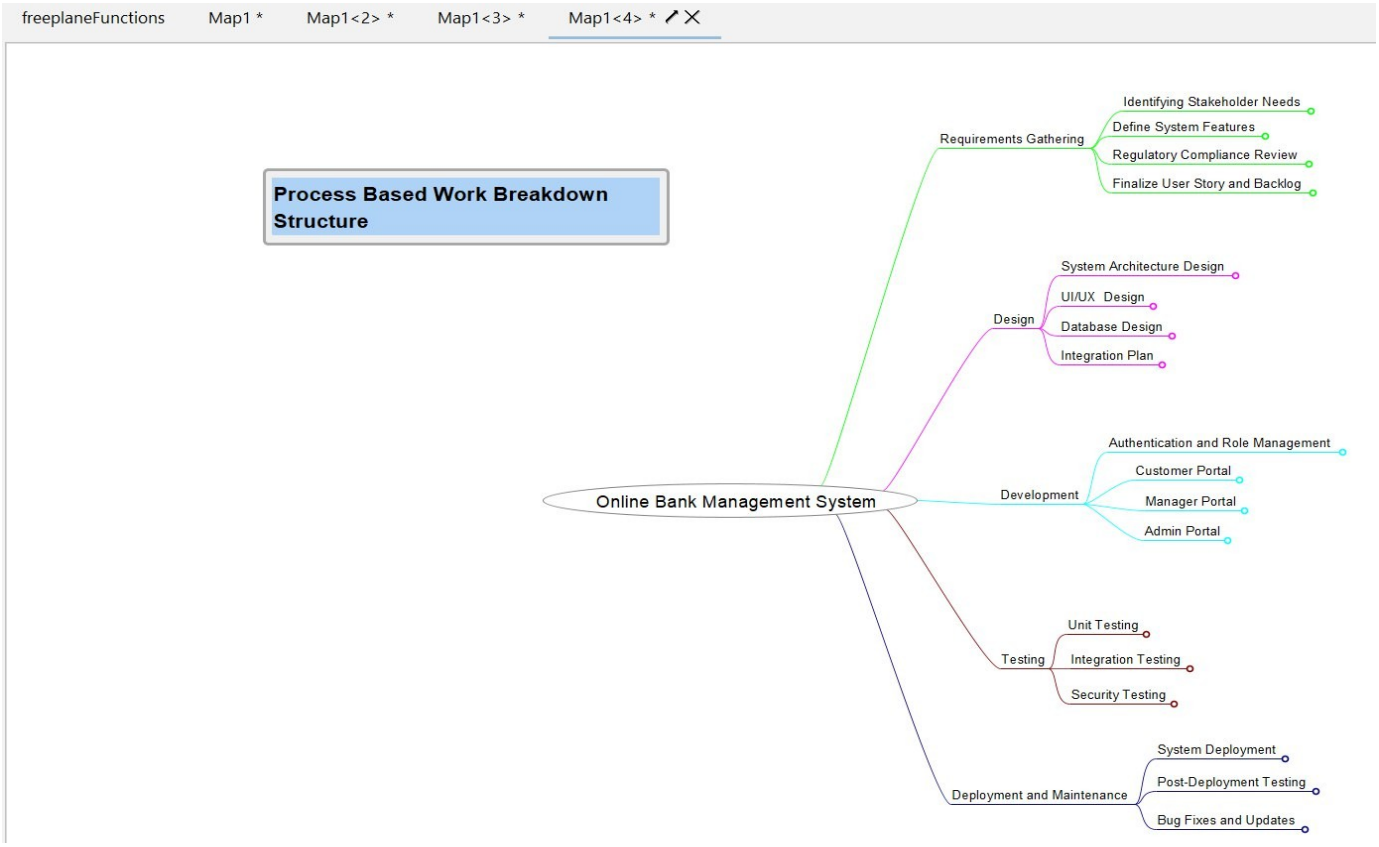
Organizes tasks based on the sequential phases of the Software Development Life Cycle (SDLC).

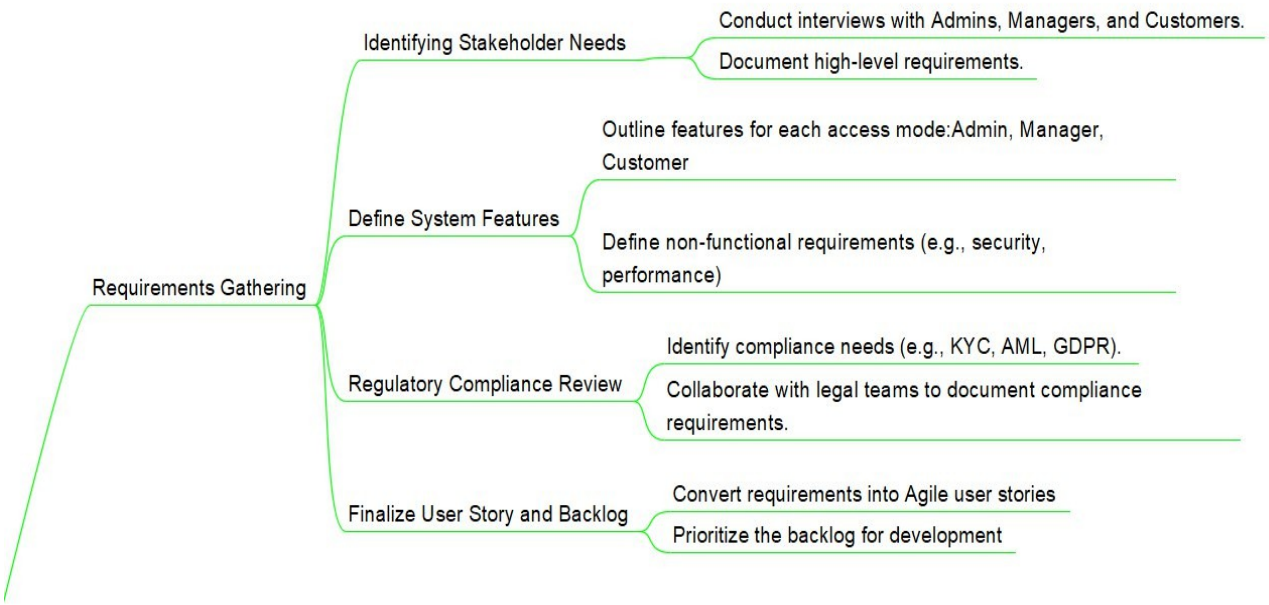
Phases and Tasks:

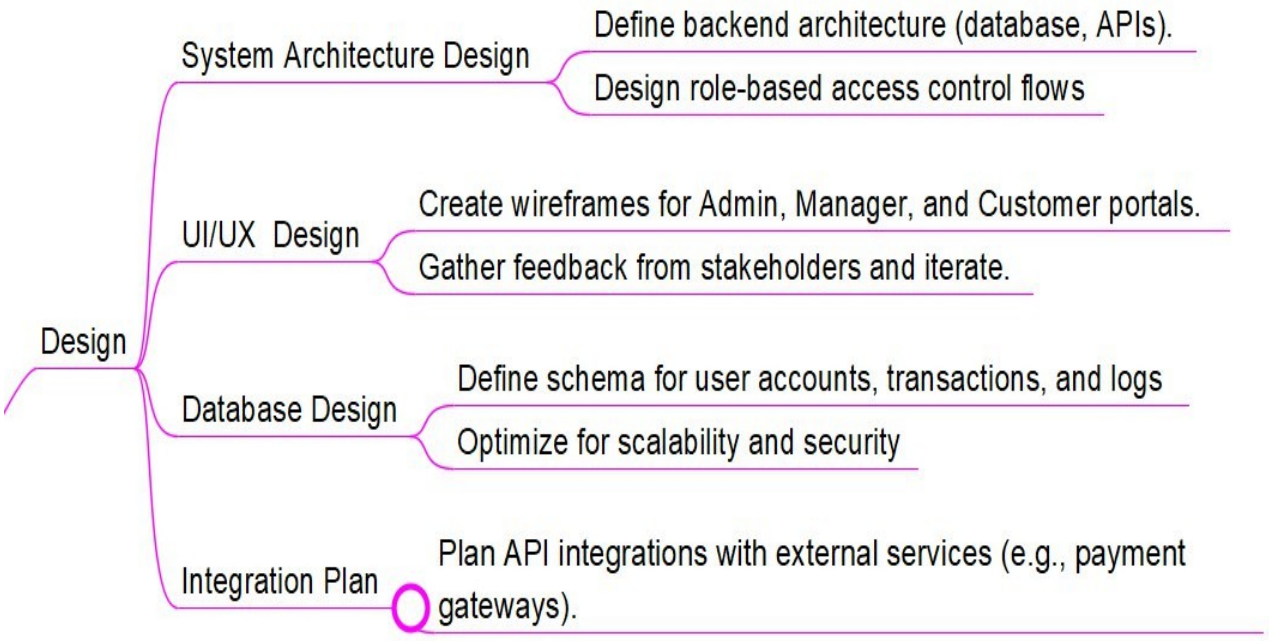
1. **Requirements Gathering (10 Days)**
    - Identify Stakeholders (2 Days): Admins, Managers, Customers.
    - Define System Features (3 Days): User roles and key functionalities.
    - Regulatory and Compliance Analysis (3 Days): Ensure compliance with AML, KYC, GDPR.
    - Create User Stories (2 Days): Define Agile user stories for sprint planning.
2. **Design (15 Days)**
    - High-Level Architecture (5 Days): Backend architecture, APIs.
    - UI/UX Design (5 Days): Wireframes for all portals.
    - Database Design (3 Days): Schema for users, transactions, and logs.
    - Integration Plan (2 Days): External system connections.
3. **Development (40 Days)**
    - Sprint 1: Authentication and Security (10 Days)
    - Sprint 2: Customer Portal (15Days)
    - Sprint 3: Manager Portal (15 Days)
    - Sprint 4: Admin Portal (15 Days)
4. **Testing (15 Days)**
    - Unit Testing (5 Days): Validate modules like login and transactions.

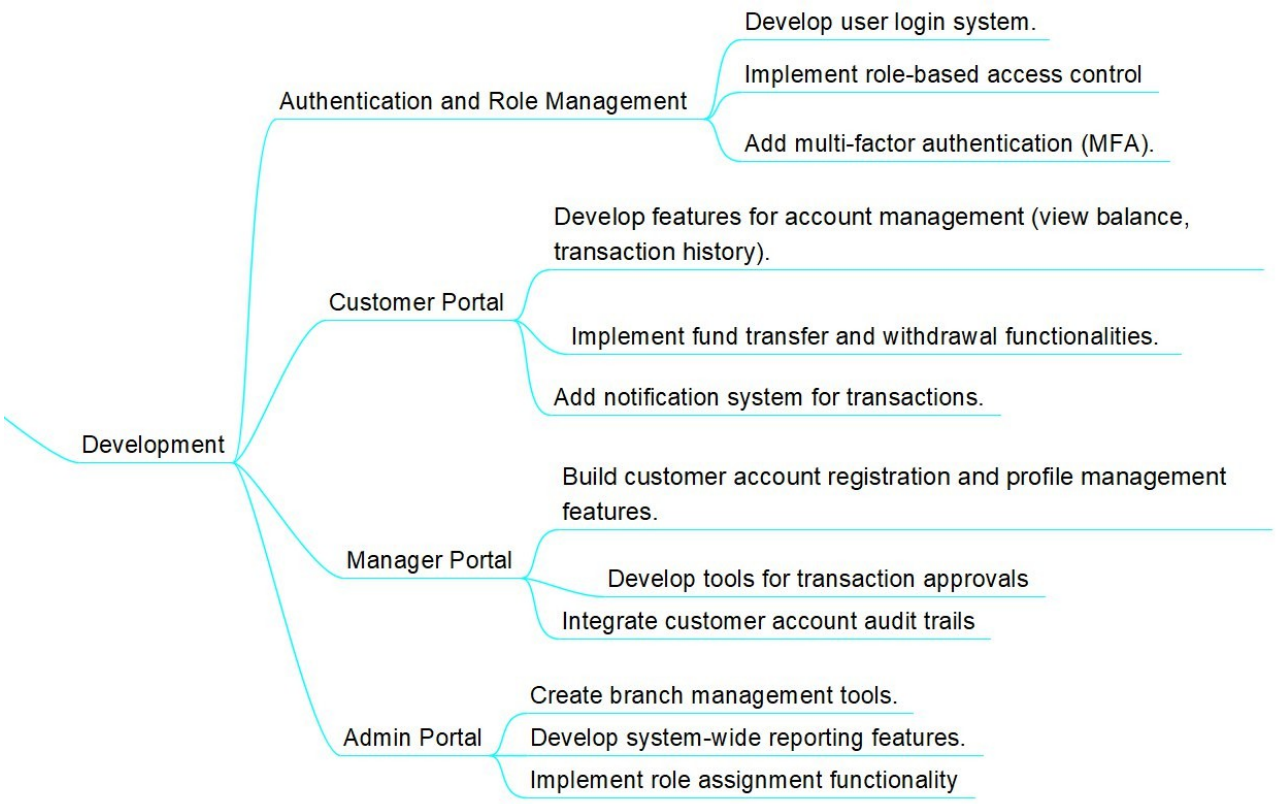
- Integration Testing (5 Days): Test interactions across workflows.
- Security Testing (5 Days): Conduct penetration tests.

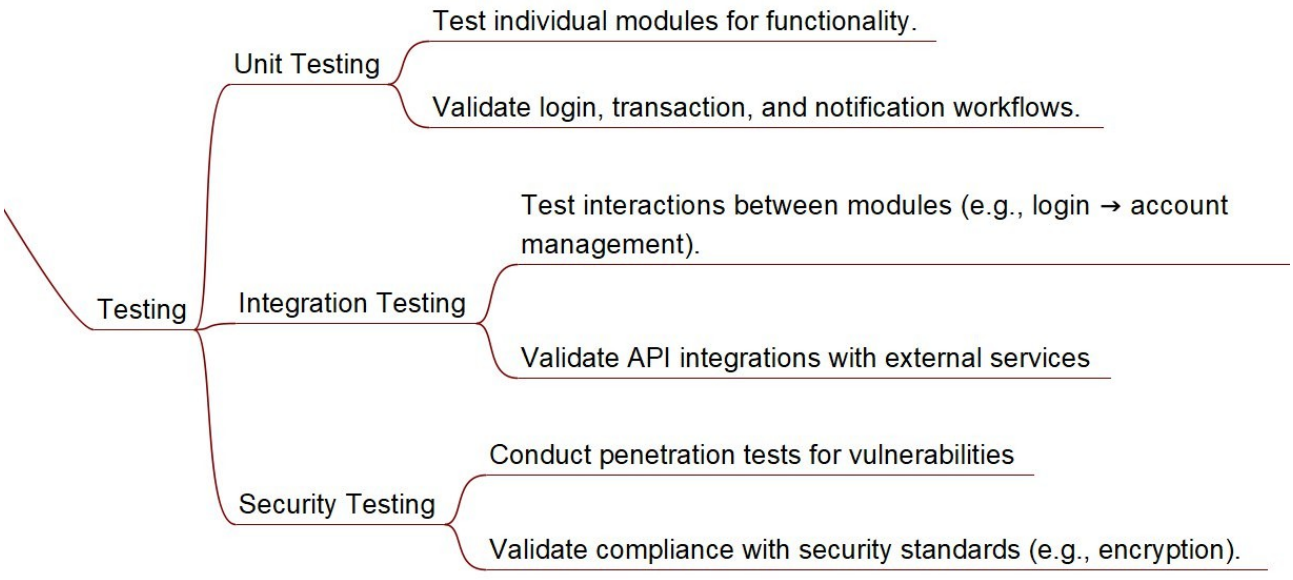
5. **Deployment and Maintenance (10 Days)**
    - Deployment (3 Days): Launch system on production servers.
    - Post-Deployment Testing (2 Days): Validate stability.
    - Bug Fixes and Updates (5 Days): Address post-launch issues

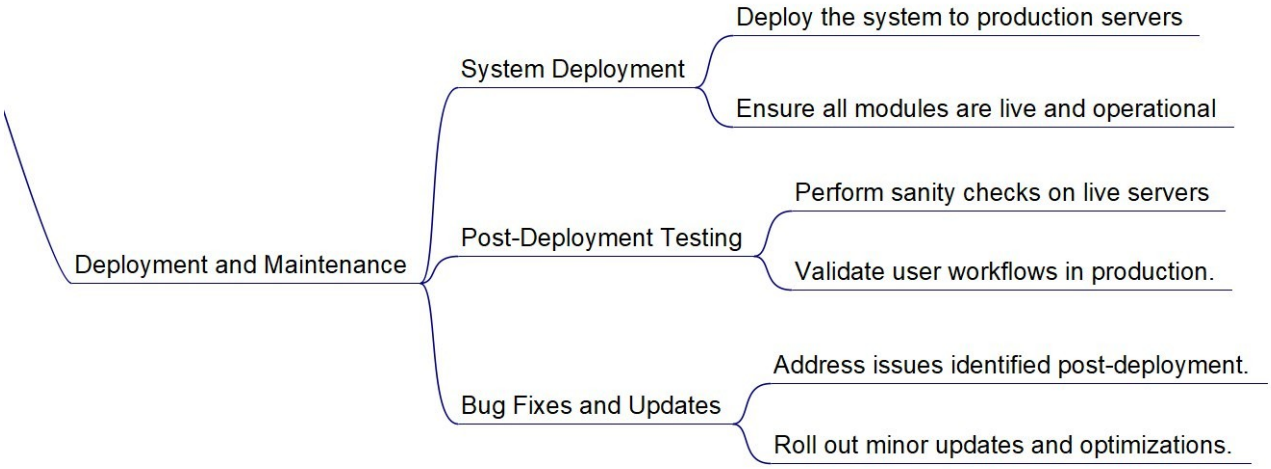



Requirements Gathering
Identifying Stakeholder Needs
Conduct interviews with Admins, Managers, and Customers.
Document high-level requirements.
Define System Features
Outline features for each access mode:Admin, Manager, Customer
Define non-functional requirements (e.g., security, performance)
Regulatory Compliance Review
Identify compliance needs (e.g., KYC, AML, GDPR).
Collaborate with legal teams to document compliance requirements.
Finalize User Story and Backlog
Convert requirements into Agile user stories
Prioritize the backlog for development

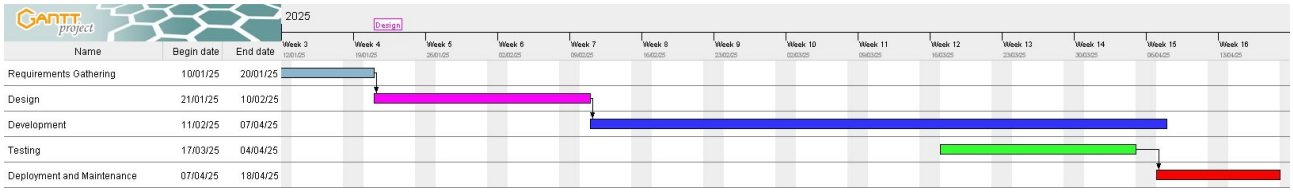

Design
System Architecture Design
Define backend architecture (database, APIs).
Design role-based access control flows
UI/UX Design
Create wireframes for Admin, Manager, and Customer portals.
Gather feedback from stakeholders and iterate.
Database Design
Define schema for user accounts, transactions, and logs
Optimize for scalability and security
Integration Plan
Plan API integrations with external services (e.g., payment gateways).

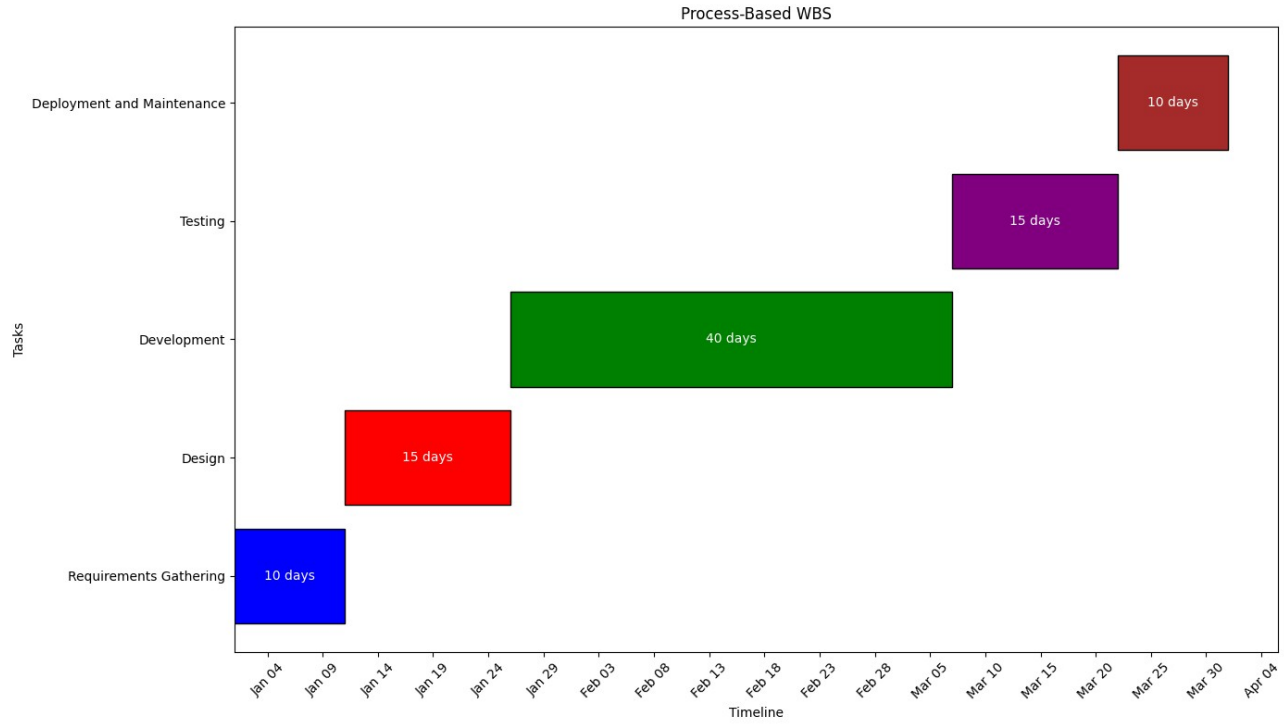

Development
Authentication and Role Management
Develop user login system.
Implement role-based access control
Add multi-factor authentication (MFA).
Customer Portal
Develop features for account management (view balance, transaction history).
Implement fund transfer and withdrawal functionalities.
Add notification system for transactions.
Manager Portal
Build customer account registration and profile management features.
Develop tools for transaction approvals
Integrate customer account audit trails
Admin Portal
Create branch management tools.
Develop system-wide reporting features.
Implement role assignment functionality


Testing
Unit Testing
Test individual modules for functionality.
Validate login, transaction, and notification workflows.
Integration Testing
Test interactions between modules (e.g., login → account management).
Validate API integrations with external services
Security Testing
Conduct penetration tests for vulnerabilities
Validate compliance with security standards (e.g., encryption).


Deployment and Maintenance
System Deployment
Deploy the system to production servers
Ensure all modules are live and operational
Post-Deployment Testing
Perform sanity checks on live servers
Validate user workflows in production.
Bug Fixes and Updates
Address issues identified post-deployment.
Roll out minor updates and optimizations.

Timeline:


2025
Design
Name
Begin date
End date
Requirements Gathering 10/01/25 20/01/25
Design 21/01/25 10/02/25
Development 11/02/25 07/04/25
Testing 17/03/25 04/04/25
Deployment and Maintenance 07/04/25 18/04/25


Process-Based WBS
Deployment and Maintenance
Testing
Development
Design
Requirements Gathering
10 days
15 days
40 days
15 days
10 days
Tasks
Jan 04
Jan 09
Jan 14
Jan 19
Jan 24
Jan 29
Feb 03
Feb 08
Feb 13
Feb 18
Feb 23
Feb 28
Mar 05
Mar 10
Mar 15
Mar 20
Mar 25
Mar 30
Apr 04
Timeline

Process-Based WBS Dependencies

1. **Design depends on Requirements Gathering**:
    - The design phase can only begin after stakeholders' needs and system features are fully defined.
2. **Development depends on Design**:
    - Development tasks rely on finalized architecture, UI/UX designs, and database schemas.
3. **Testing depends on Development**:
    - Testing can only start after core modules and workflows are implemented.
4. **Deployment depends on Testing**:
    - Deployment occurs only after successful completion of all testing phases.

2. Product-Based WBS

Organizes tasks based on deliverables or system components.

Components and Tasks Divided by Categories:

A. User Interfaces (30 Days)

1. **Customer Portal (15 Days)**
    - Design (4 Days): Create wireframes for login, balance inquiry, and transactions.
    - Develop (8 Days): Build the frontend using appropriate frameworks.
    - Test (3 Days): Validate functionality and responsiveness.
2. **Manager Portal (15 Days)**
    - Design (4 Days): Create interfaces for account approvals and management.
    - Develop (8 Days): Build pages for registration and transaction reviews.
    - Test (3 Days): Conduct UI testing.
3. **Admin Portal (15 Days)**
    - Design (4 Days): Create dashboards for branch and role management.
    - Develop (8 Days): Build analytics and reporting interfaces.
    - Test (3 Days): Validate admin workflows.

B. Backend Modules (40 Days)

1. **Authentication and Security (15 Days)**
    - Role-Based Login System (5 Days): Implement authentication logic.
    - Multi-Factor Authentication (4 Days): Add OTP/email verification.

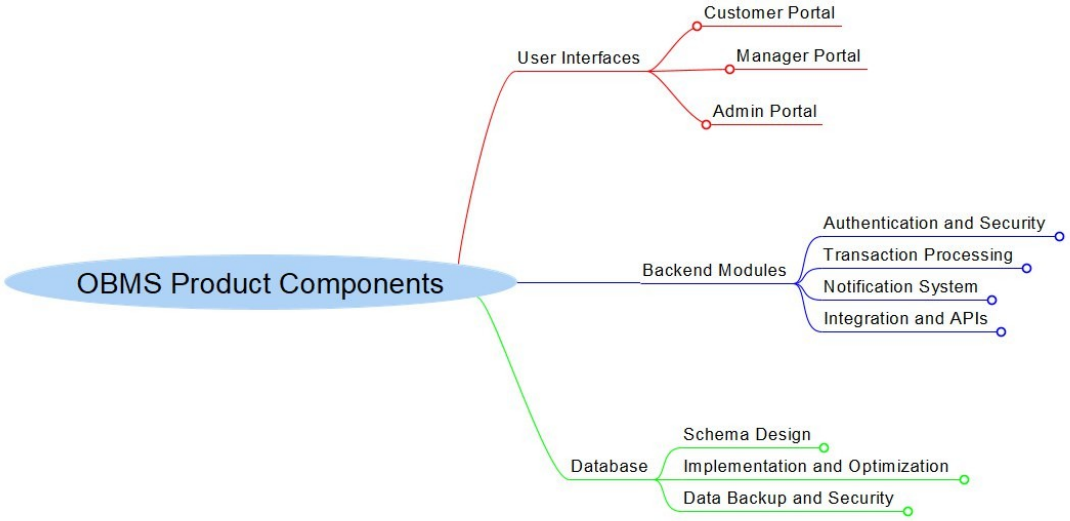
- Encryption (3 Days): Secure sensitive data.
- Penetration Testing (3 Days): Conduct security audits.

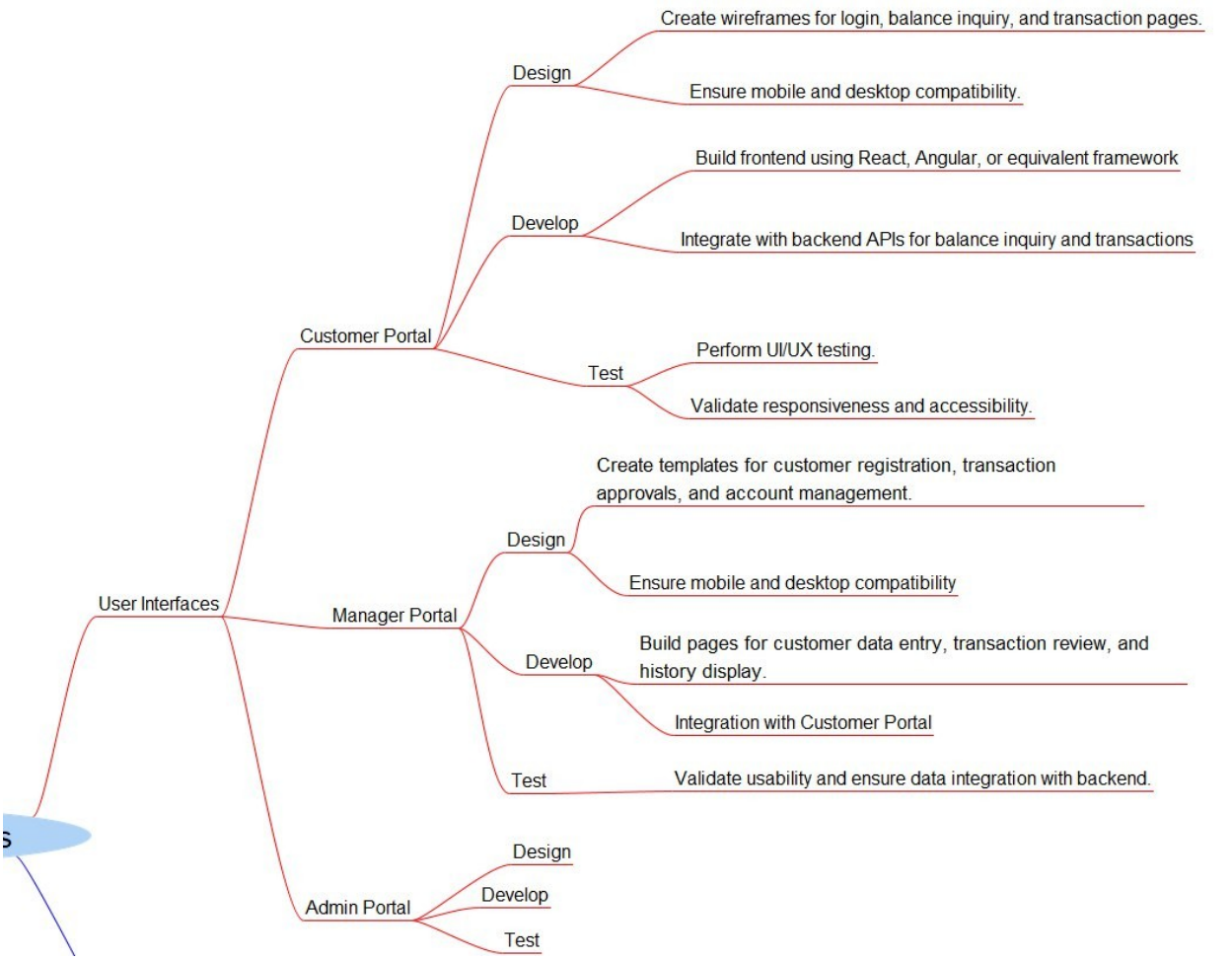
2. **Transaction Processing (15 Days)**
    - Develop Fund Transfers (7 Days): Internal and external workflows.
    - Withdrawals and Deposits (5 Days): Simulate operations.
    - Logging Mechanisms (3 Days): Ensure robust transaction records.
3. **Notifications (10 Days)**
    - Real-Time Alerts (6 Days): Implement SMS and email notifications.
    - System Alerts (4 Days): Add alerts for admin users.

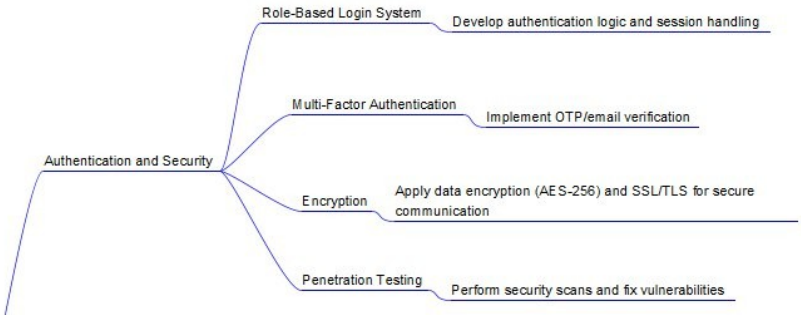
C. Database (20 Days)

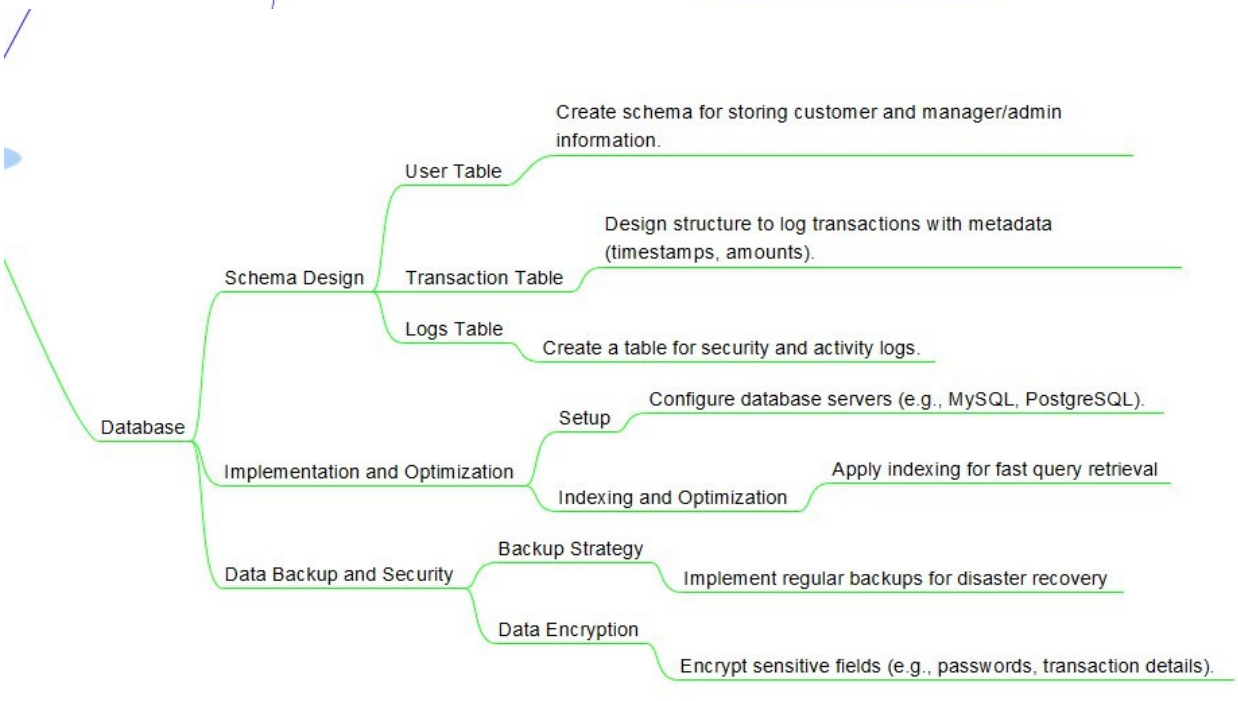
1. **Schema Design (10 Days)**
    - User Table (3 Days): Structure for user accounts.
    - Transaction Table (5 Days): Detailed schema for logging transactions.
    - Logs Table (2 Days): Security and activity logs.
2. **Implementation and Optimization (5 Days)**
    - Configure Databases (3 Days): Set up environments.
    - Optimize Indexing (2 Days): Ensure efficient queries.
3. **Backup and Recovery (5 Days)**
    - Backup Strategy (3 Days): Regular backups for disaster recovery.
    - Recovery Protocols (2 Days): Test and validate recovery processes.

Product Based Work Break Down Structure

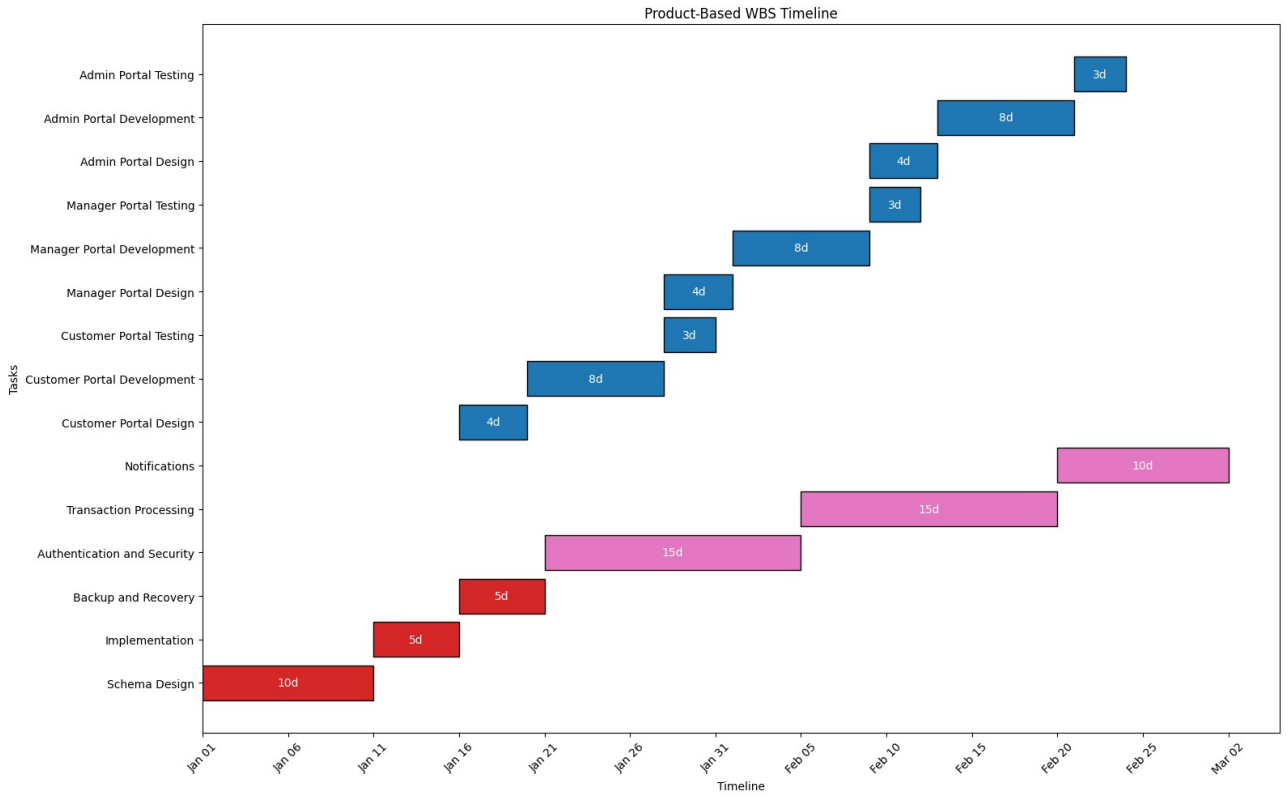



User Interfaces
Customer Portal
Design
Create wireframes for login, balance inquiry, and transaction pages.
Ensure mobile and desktop compatibility.
Develop
Build frontend using React, Angular, or equivalent framework
Integrate with backend APIs for balance inquiry and transactions
Test
Perform UI/UX testing.
Validate responsiveness and accessibility.
Manager Portal
Design
Create templates for customer registration, transaction approvals, and account management.
Ensure mobile and desktop compatibility
Develop
Build pages for customer data entry, transaction review, and history display.
Integration with Customer Portal
Test
Validate usability and ensure data integration with backend.
Admin Portal
Design
Develop
Test


Authentication and Security
Role-Based Login System
Develop authentication logic and session handling
Multi-Factor Authentication
Implement OTP/email verification
Encryption
Apply data encryption (AES-256) and SSL/TLS for secure communication
Penetration Testing
Perform security scans and fix vulnerabilities


Database
Schema Design
User Table
Create schema for storing customer and manager/admin information.
Transaction Table
Design structure to log transactions with metadata (timestamps, amounts).
Logs Table
Create a table for security and activity logs.
Implementation and Optimization
Setup
Configure database servers (e.g., MySQL, PostgreSQL).
Indexing and Optimization
Apply indexing for fast query retrieval
Data Backup and Security
Backup Strategy
Implement regular backups for disaster recovery
Data Encryption
Encrypt sensitive fields (e.g., passwords, transaction details).

Timeline:



Product-Based WBS Dependencies

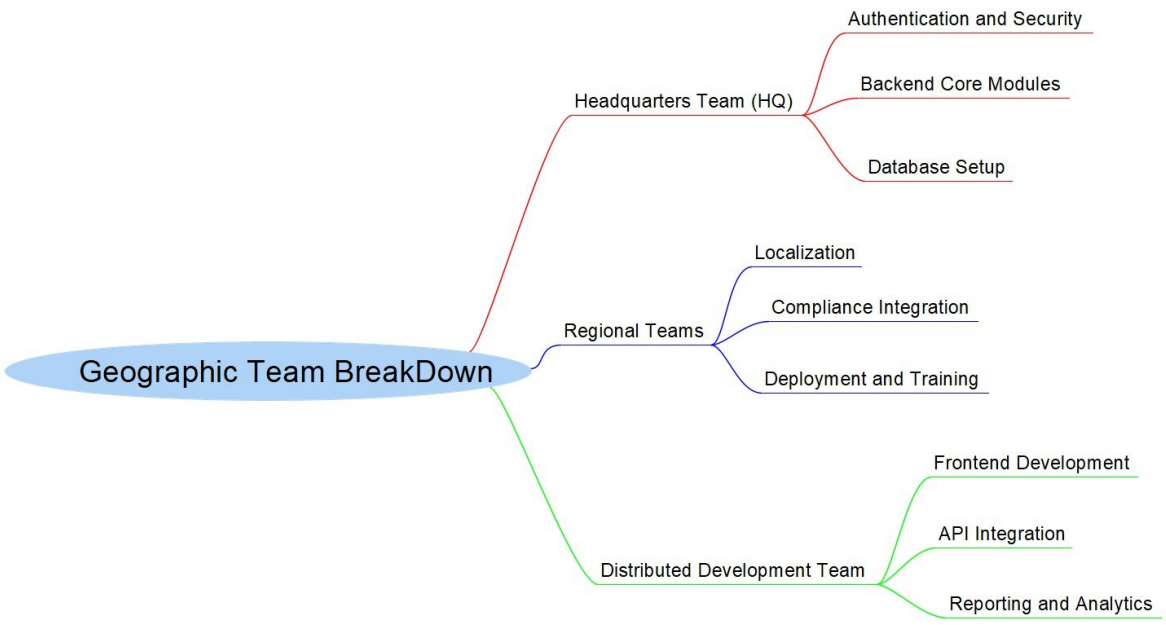
1. **User Interfaces depend on Backend Modules**:
    - UI tasks like Customer Portal development require backend APIs to be functional.
2. **Notifications depend on Transaction Processing**:
    - Real-time alerts need data from transaction workflows.
3. **Database tasks influence Backend Modules**:
    - Backend logic depends on schema design and database readiness.
4. **Backend Modules influence User Interfaces and Notifications**:
    - Core authentication and transaction features must be operational for UI and notification systems.

3. Geographic-Based WBS

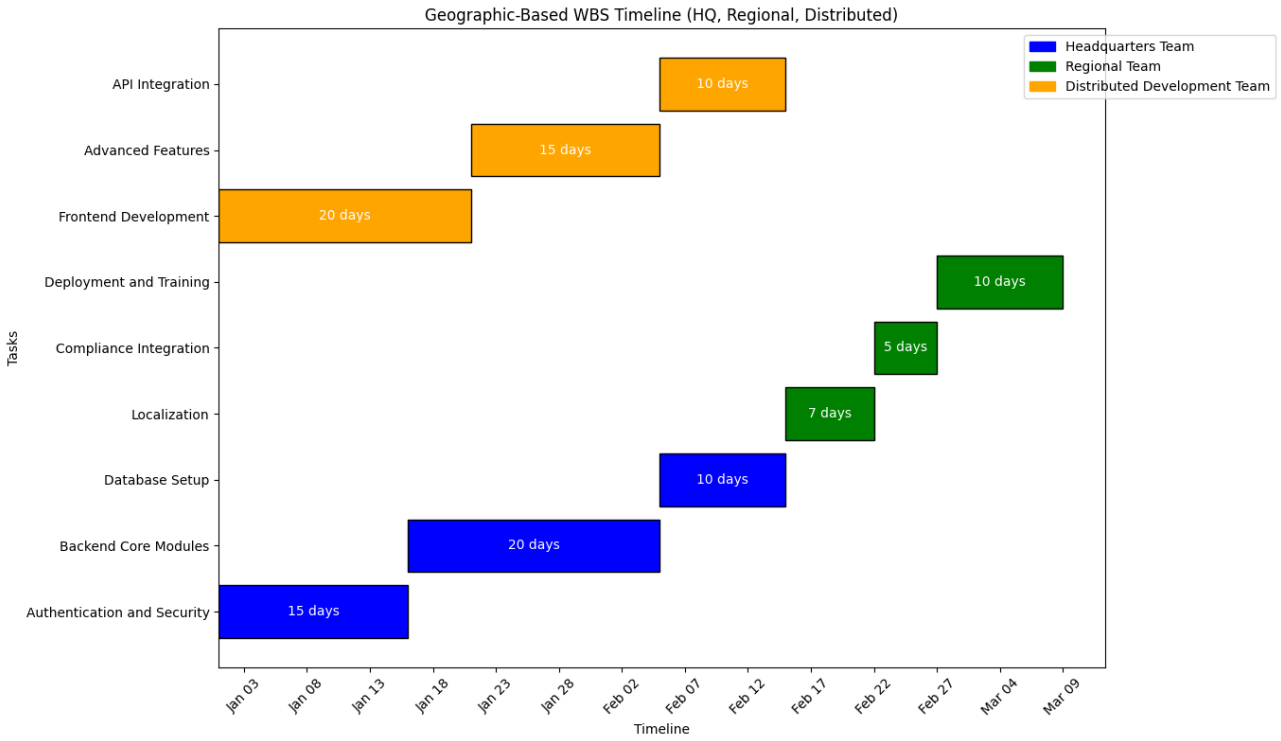
Organizes tasks by team location and regional implementation.

Team Locations and Tasks:

1. **Headquarters Team (45 Days)**
    - Authentication and Security (15 Days): Develop secure login.
    - Backend Core Modules (20 Days): Transactions and notifications.
    - Database Setup (10 Days): Schema design and indexing.
2. **Regional Teams (22 Days per Region){Overlaps}**
    - Localization (7 Days): Translate UI and workflows.
    - Compliance Integration (5 Days): Align with regional regulations.
    - Deployment and Training (10 Days): Launch system and train staff.
3. **Distributed Development Teams (45 Days)**
    - Frontend Development (20 Days): Customer, Manager, and Admin portals.
    - Advanced Features (15 Days): Fraud detection and dashboards.
    - API Integration (10 Days): Payment gateways and external systems.


Geographic Team BreakDown
Headquarters Team (HQ)
Authentication and Security
Backend Core Modules
Database Setup
Regional Teams
Localization
Compliance Integration
Deployment and Training
Distributed Development Team
Frontend Development
API Integration
Reporting and Analytics

Timeline:


Geographic-Based WBS Timeline (HQ, Regional, Distributed)
Headquarters Team
Regional Team
Distributed Development Team
API Integration
Advanced Features
Frontend Development
Deployment and Training
Compliance Integration
Localization
Database Setup
Backend Core Modules
Authentication and Security
10 days
15 days
20 days
10 days
5 days
7 days
10 days
20 days
15 days
Tasks
Jan 03
Jan 08
Jan 13
Jan 18
Jan 23
Jan 28
Feb 02
Feb 07
Feb 12
Feb 17
Feb 22
Feb 27
Mar 04
Mar 09
Timeline

Geographic-Based WBS Dependencies

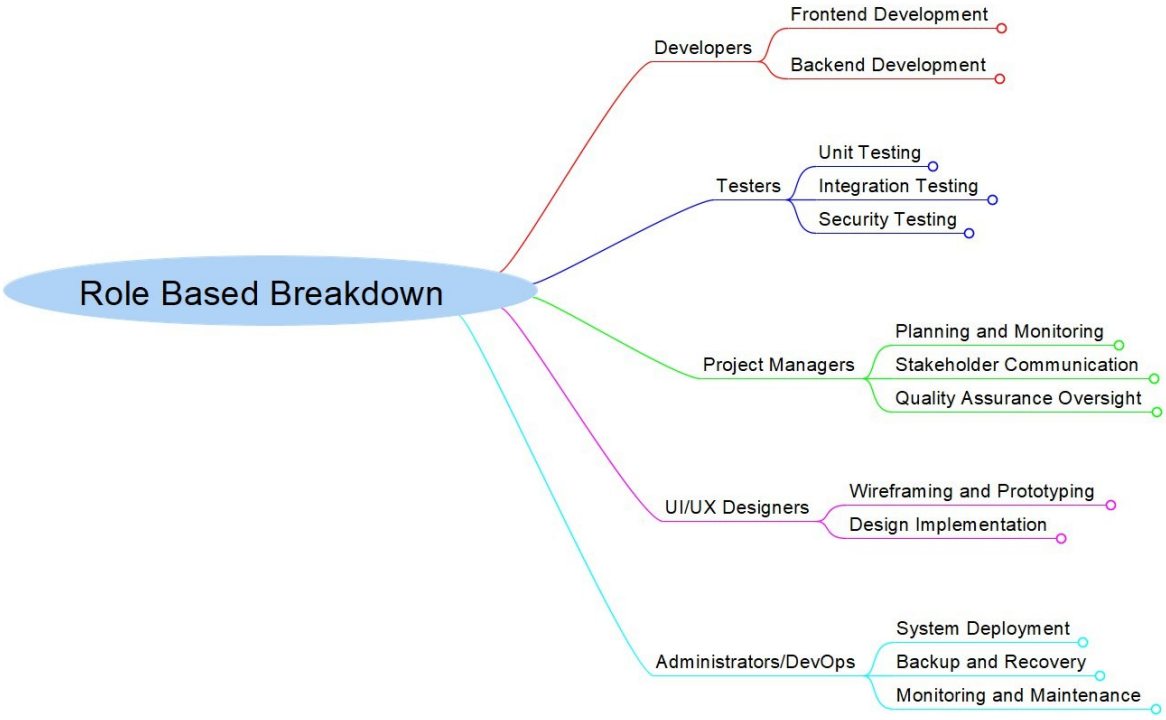
1. **Regional Teams depend on Headquarters Team**:
    - Localization, compliance integration, and deployment tasks rely on HQ completing Authentication, Backend Core Modules, and Database Setup.
2. **Distributed Teams depend on Backend APIs from HQ**:
    - Frontend development and API integration tasks require backend readiness.
3. **Deployment depends on Localization and Compliance**:
    - Regional deployment starts only after local compliance and translations are validated.

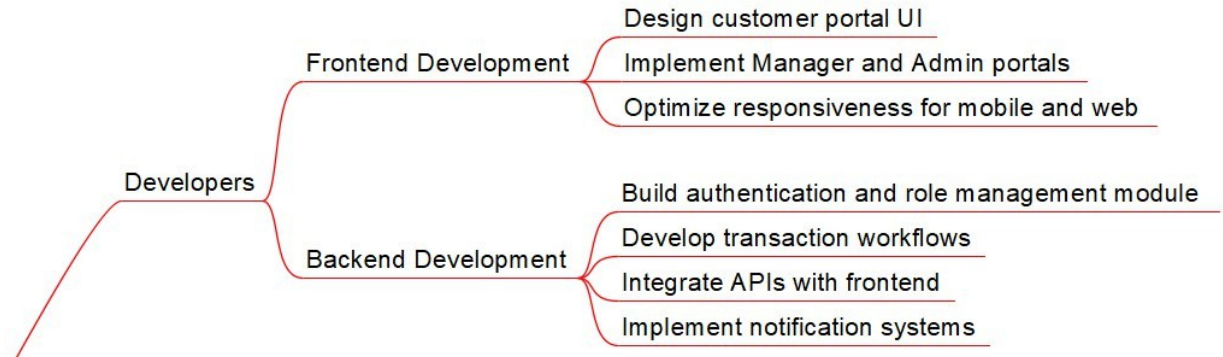
4. Role-Based WBS

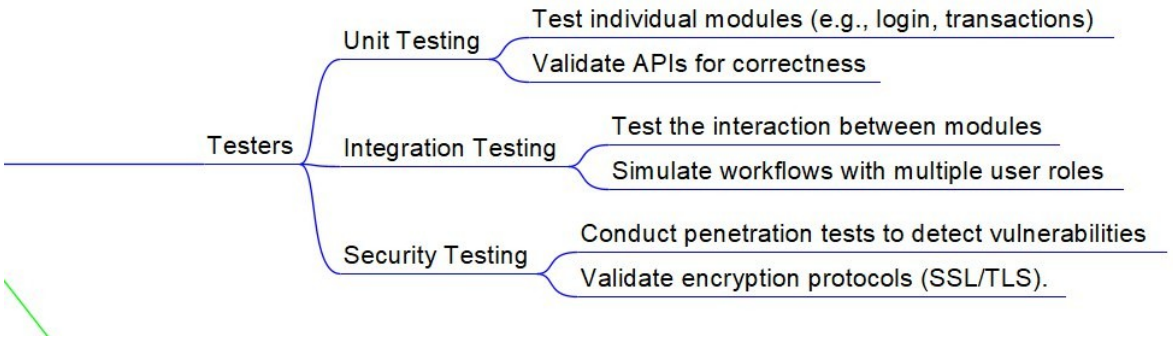
Organizes tasks by roles in the project.

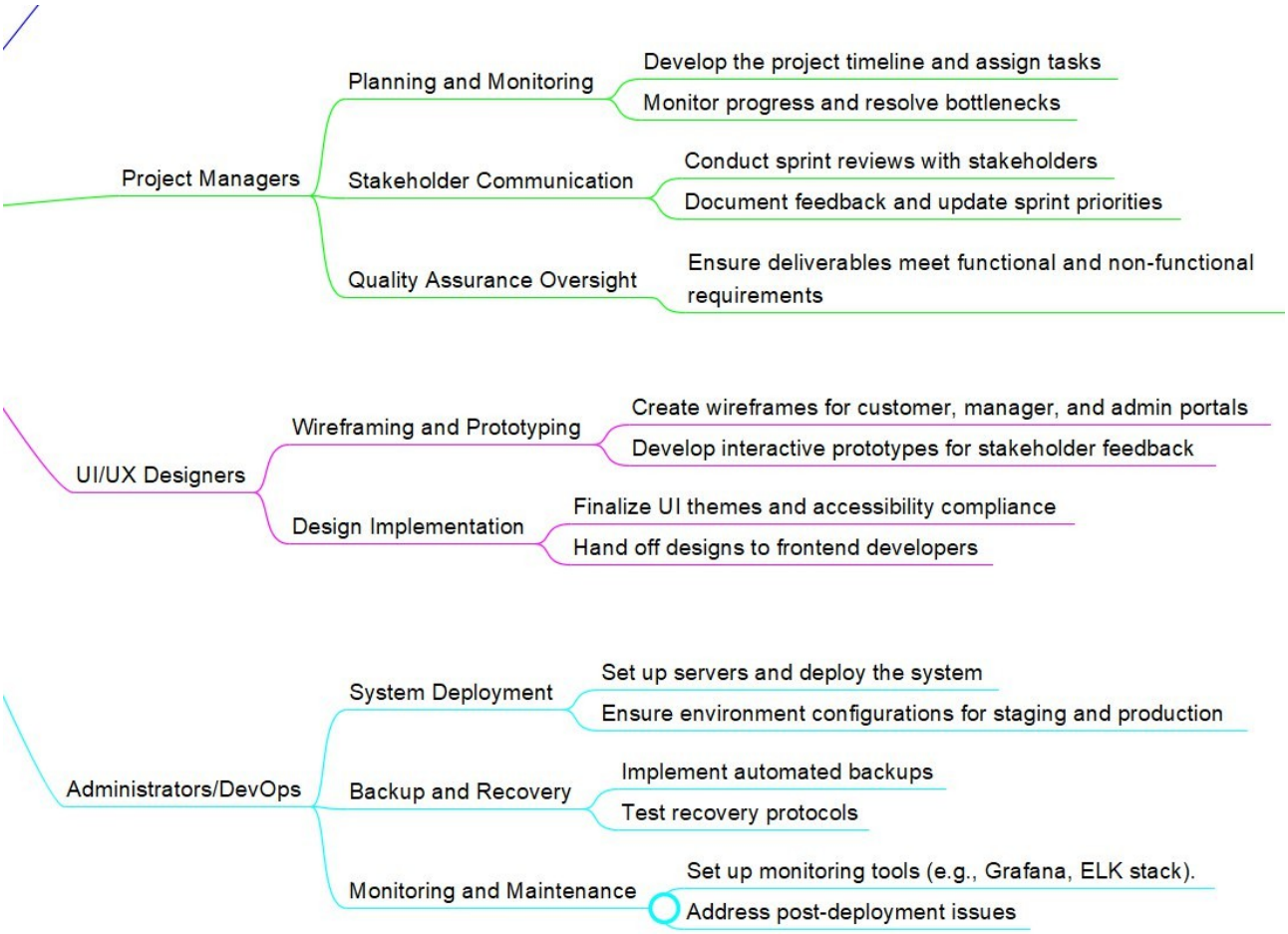
Roles and Tasks:

1. **Developers (30 Days)**
    - Frontend Development (20 Days): Implement portals.
    - Backend Development (30 Days): Build authentication, APIs, and notifications.
2. **Testers (15 Days)**
    - Unit Testing (5 Days): Validate individual modules.
    - Integration Testing (5 Days): Simulate workflows.
    - Security Testing (5 Days): Penetration tests and encryption validation.
3. **Project Managers (10 Days)**
    - Planning and Monitoring (8 Days): Develop timeline and track progress.
    - Stakeholder Communication (2 Days): Sprint reviews and feedback.
4. **UI/UX Designers (15 Days)**
    - Wireframing and Prototyping (10 Days): Create interactive wireframes.
    - Design Implementation (7 Days): Finalize themes and layouts.
5. **Administrators/DevOps (20Days)**
    - Deployment (10 Days): Set up servers.
    - Backup and Recovery (5 Days): Implement disaster recovery.
    - Monitoring and Maintenance (5 Days): Post-deployment monitoring.

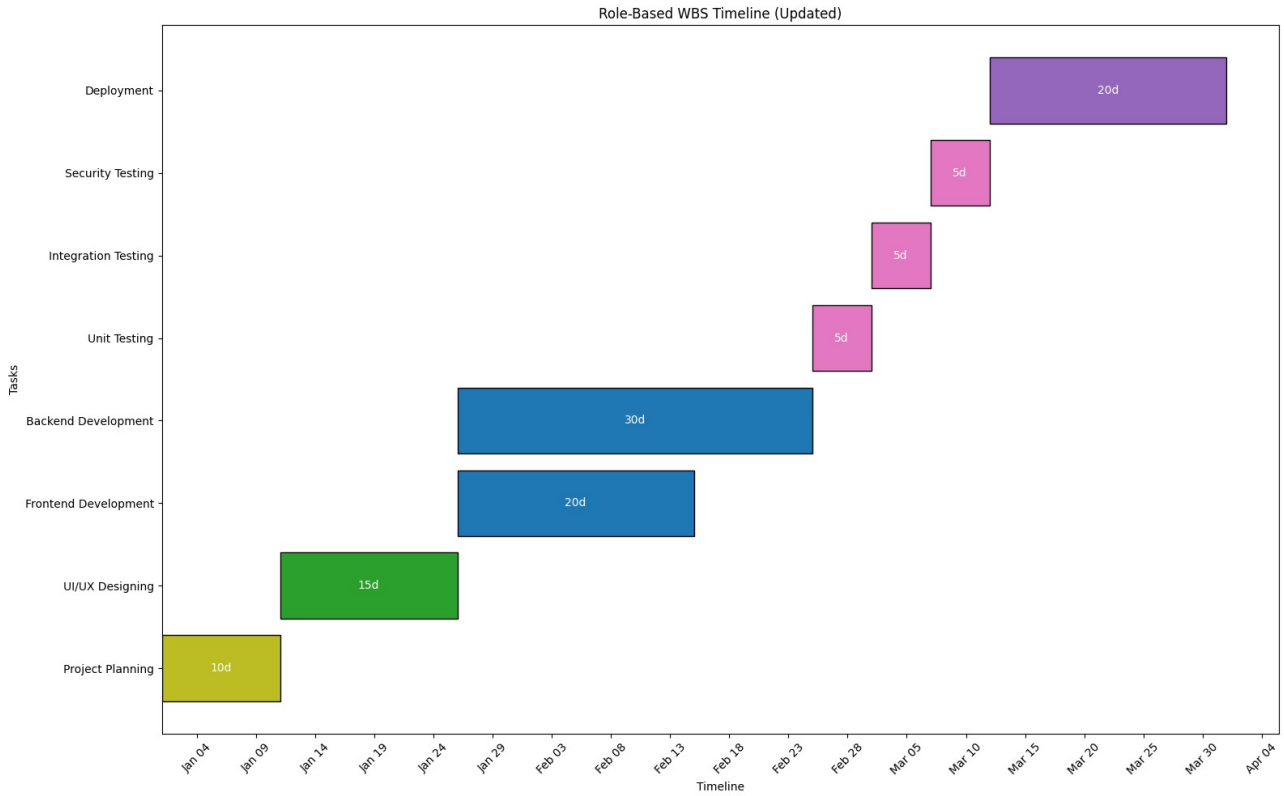

Role Based Breakdown
Developers
Frontend Development
Backend Development
Testers
Unit Testing
Integration Testing
Security Testing
Project Managers
Planning and Monitoring
Stakeholder Communication
Quality Assurance Oversight
UI/UX Designers
Wireframing and Prototyping
Design Implementation
Administrators/DevOps
System Deployment
Backup and Recovery
Monitoring and Maintenance


Developers
Frontend Development
Design customer portal UI
Implement Manager and Admin portals
Optimize responsiveness for mobile and web
Backend Development
Build authentication and role management module
Develop transaction workflows
Integrate APIs with frontend
Implement notification systems


Testers
Unit Testing
Test individual modules (e.g., login, transactions)
Validate APIs for correctness
Integration Testing
Test the interaction between modules
Simulate workflows with multiple user roles
Security Testing
Conduct penetration tests to detect vulnerabilities
Validate encryption protocols (SSL/TLS).


Project Managers
Planning and Monitoring
Develop the project timeline and assign tasks
Monitor progress and resolve bottlenecks
Stakeholder Communication
Conduct sprint reviews with stakeholders
Document feedback and update sprint priorities
Quality Assurance Oversight
Ensure deliverables meet functional and non-functional requirements
UI/UX Designers
Wireframing and Prototyping
Create wireframes for customer, manager, and admin portals
Develop interactive prototypes for stakeholder feedback
Design Implementation
Finalize UI themes and accessibility compliance
Hand off designs to frontend developers
Administrators/DevOps
System Deployment
Set up servers and deploy the system
Ensure environment configurations for staging and production
Backup and Recovery
Implement automated backups
Test recovery protocols
Monitoring and Maintenance
Set up monitoring tools (e.g., Grafana, ELK stack).
Address post-deployment issues

Timeline:


Role-Based WBS Timeline (Updated)
Tasks
Deployment
Security Testing
Integration Testing
Unit Testing
Backend Development
Frontend Development
UI/UX Designing
Project Planning
20d
5d
5d
5d
30d
20d
15d
10d
Jan 04
Jan 09
Jan 14
Jan 19
Jan 24
Jan 29
Feb 03
Feb 08
Feb 13
Feb 18
Feb 23
Feb 28
Mar 05
Mar 10
Mar 15
Mar 20
Mar 25
Mar 30
Apr 04
Timeline

Role-Based WBS Dependencies

1. **Testers depend on Developers**:
    - Testing tasks like unit and integration testing rely on modules developed by the developers.
2. **UI/UX Designers influence Developers**:
    - Developers require finalized designs for frontend implementation.
3. **Administrators/DevOps depend on Developers and Testers**:
    - Deployment and monitoring tasks require developed and tested modules.
4. **Project Managers oversee all phases**:
    - Effective planning and monitoring ensure that dependencies across all roles are managed and resolved.

Unified Timeline Summary

WBS Type	Key Focus	Aligned Duration
Process-Based	Sequential SDLC Phases	90 Days
Product-Based	Functional Components	90 Days
Geographic-Based	Team Location	90 Days
Role-Based	Task Assignment by Role	90 Days

Python Code for generating Timelines:

```python
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from datetime import datetime, timedelta

def create_gantt_chart_with_durations(tasks, title, legend_patches=None):
    fig, ax = plt.subplots(figsize=(14, 8))
    task_names = [task["task"] for task in tasks]
    start_dates = [mdates.date2num(task["start_date"]) for task in tasks]
    durations = [task["duration"] for task in tasks]
    colors = [task["color"] for task in tasks]

    # Plot tasks with durations displayed
    for i, task in enumerate(tasks):
        ax.barh(task_names[i], durations[i], left=start_dates[i], color=colors[i], edgecolor="black")
        # Add duration label on the bar
        mid_point = start_dates[i] + (durations[i] / 2)
        ax.text(mid_point, i, f"{durations[i]} days", va='center', ha='center', color="white",
fontsize=10)

    # Configure the chart
    ax.xaxis.set_major_formatter(mdates.DateFormatter("%b %d"))
    ax.xaxis.set_major_locator(mdates.DayLocator(interval=5))
    plt.xticks(rotation=45)
    ax.set_xlabel("Timeline")
    ax.set_ylabel("Tasks")
    ax.set_title(title)

    # Add legend if provided
    if legend_patches:
        plt.legend(handles=legend_patches, loc='upper right', bbox_to_anchor=(1.2, 1.0))

    plt.tight_layout()
    plt.show()

# Convert days to datetime within a 90-day project timeline
start_date = datetime(2025, 1, 1)

def calculate_dates(start_offset, duration):
    start_date = datetime(2025, 1, 1) + timedelta(days=start_offset)
    return start_date, duration
```

```python
import matplotlib.patches as mpatches


geographic_based_tasks = [
    # Headquarters Team Tasks
    {"task": "Authentication and Security", "start_date": calculate_dates(0, 15)[0], "duration": 15,
"color": "blue"},
    {"task": "Backend Core Modules", "start_date": calculate_dates(15, 20)[0], "duration": 20, "color":
"blue"},
    {"task": "Database Setup", "start_date": calculate_dates(35, 10)[0], "duration": 10, "color":
"blue"},

    # Regional Team Tasks
    {"task": "Localization", "start_date": calculate_dates(45, 7)[0], "duration": 7, "color": "green"},
    {"task": "Compliance Integration", "start_date": calculate_dates(52, 5)[0], "duration": 5, "color":
"green"},
    {"task": "Deployment and Training", "start_date": calculate_dates(57, 10)[0], "duration": 10,
"color": "green"},

    # Distributed Development Team Tasks
    {"task": "Frontend Development", "start_date": calculate_dates(0, 20)[0], "duration": 20, "color":
"orange"},
    {"task": "Advanced Features", "start_date": calculate_dates(20, 15)[0], "duration": 15, "color":
"orange"},
    {"task": "API Integration", "start_date": calculate_dates(35, 10)[0], "duration": 10, "color":
"orange"},
]
```

```
# 4. Role-Based WBS
role_based_tasks = [
    {"task": "Project Managers: Planning", "start_date": calculate_dates(0, 15)[0], "duration": 15,
"color": "cyan"},  # Starts on Day 0
    {"task": "UI/UX Designers: Wireframes", "start_date": calculate_dates(15, 10)[0], "duration": 10,
"color": "lime"},  # Starts after Project Planning
    {"task": "Developers: Frontend", "start_date": calculate_dates(25, 20)[0], "duration": 20, "color":
"blue"},  # Starts after UI/UX Designers finish
    {"task": "Developers: Backend", "start_date": calculate_dates(25, 30)[0], "duration": 30, "color":
"orange"},  # Backend development can start independently
    {"task": "Testers: Unit Testing", "start_date": calculate_dates(55, 5)[0], "duration": 5, "color":
"green"},
    {"task": "Testers: Integration Testing", "start_date": calculate_dates(60, 5)[0], "duration": 5,
"color": "purple"},
    {"task": "Testers: Security Testing", "start_date": calculate_dates(65, 5)[0], "duration": 5,
"color": "red"},
    {"task": "Administrators: Deployment", "start_date": calculate_dates(70, 7)[0], "duration": 7,
"color": "gold"},
]



# Generate Gantt chart with color-coded bars
create_gantt_chart_with_durations(geographic_based_tasks, "Geographic-Based WBS Timeline (HQ, Regional,
Distributed)",legend_patches=legend_patches)


# Generate Gantt charts
create_gantt_chart_with_durations(process_based_tasks, "Process-Based WBS")
```

--Thank You--