Table of Contents

***~~Table of Contents 2~~***

***~~1. INTRODUCTION TO WASL SYSTEM 4~~***

***~~1.1 Overview (Of the Problem) 4~~***

***~~1.2 Purpose (Of Your Solution/Software) 4~~***

***~~1.3 Scope (Domain and Main Focus of the Solution) 4~~***

***~~1.4 Definitions, Acronyms, and Abbreviations 5~~***

***~~1.5 References (Dependencies, Policies, Regulations) 5~~***

***~~2. ARCHITECTURAL REPRESENTATIONS: 4+1 VIEW MODEL OVERVIEW 6~~***

***~~2.1 Introduction to the 4+1 View Model 6~~***

***~~3. LOGICAL VIEW 7~~***

***~~3.1 Overview 7~~***

***~~3.2 Class Diagram 8~~***

***~~3.2.1 Core Entity Classes: 9~~***

***~~3.2.2 Request and Session Management: 9~~***

***~~3.2.3 Supporting Service Classes: 10~~***

***~~3.2.4 Data Types and Enumerations: 10~~***

***~~3.2.5 Relationships and Interactions: 10~~***

***~~3.3 Collaboration Diagrams 10~~***

***~~3.3.1 Main Interaction Flow: 11~~***

***~~3.4 Sequence Diagrams 12~~***

***~~3.4.1 Main Sequence Flow: 13~~***

***~~4. PROCESS VIEW 13~~***

***~~4.1 Overview 13~~***

***~~4.2 Activity Diagram 14~~***

***~~5. DEVELOPMENT VIEW 15~~***

***~~5.1 Overview 16~~***

***~~5.2 Component Diagram 17~~***

***~~6. PHYSICAL VIEW 17~~***

***~~6.1 Overview 17~~***

***~~6.2 Deployment Diagram 18~~***

***~~7. QUALITY ATTRIBUTES AND SCENARIOS 18~~***

***~~7.1 Important Quality Attributes Related to the Project 19~~***

***~~7.2 Quality Attribute Scenarios 19~~***

***~~7.2.1 Scenario for Security 19~~***

***~~7.2.2 Scenario for Usability 20~~***

***~~7.2.3 Scenario for Scalability 22~~***

***~~7.2.4 Scenario for Availability 23~~***

***~~8. ARCHITECTURAL STYLE: SERVICE-ORIENTED ARCHITECTURE (SOA) 23~~***

***~~8.1 Why This Style Was Chosen: 24~~***

***~~8.2 Implementation in Wasl System: 25~~***

***~~9. TECHNOLOGY STACK 26~~***

***~~10. DOMAIN MODEL 26~~***

***~~10.1 Core Entities: 26~~***

***~~11. USE CASES & SCENARIOS 26~~***

***~~11.1 UC-01: Request Visit (Family) 26~~***

***~~11.2 UC-02: Conduct Session (Prisoner & Family) 27~~***

***~~11.3 UC-03: Enroll Prisoner Biometric 27~~***

***~~11.4 UC-04: Smart Slot Recommendation 27~~***

***~~11.5 UC-05: Management Dashboard Reporting 27~~***

***~~12. USER MANUAL DRAFT 28~~***

***~~12.1 Family Member — How to Request a Visit 28~~***

***~~12.2 Prison Staff — Enroll Biometric 28~~***

***~~12.3 Admin — Dashboard & Reports 28~~***

***~~12.4 Troubleshooting (Common) 28~~***

***~~13. HYPOTHESIS TESTING & EXPERIMENTS 28~~***

***~~14. SEQUENCE DIAGRAMS 29~~***

***~~14.1 Booking → Approval → Scheduling 29~~***

***~~14.2 Session Start with Face Recognition 29~~***

***~~15. SYSTEM ARCHITECTURE OVERVIEW 30~~***

***~~16. UI DESIGN GUIDELINES 30~~***

***~~16.1 General Principles 30~~***

***~~16.2 Visual Identity 30~~***

***~~16.3 Layout & Navigation 30~~***

***~~16.4 UI Patterns & Components 31~~***

***~~16.5 Accessibility & Inclusivity 31~~***

***~~17. AGILE FEATURE BREAKDOWN 31~~***

***~~18. DEPLOYMENT PLAN 31~~***

***~~18.1 Deployment Environments 31~~***

***~~18.2 Microservices Breakdown on Railway 32~~***

***~~18.3 Deployment Workflow 32~~***

***~~18.4 Mobile Strategy (Expo Go Only) 32~~***

***~~18.5 Pre-Deployment Checklist 32~~***

***~~18.6 Post-Deployment Steps 32~~***

***~~19. Member’s Task Breakdown 33~~***

1. INTRODUCTION TO WASL SYSTEM

1.1 Overview (Of the Problem)

The traditional method of conducting in-person visits between prisoners and their families poses several challenges. Family members often endure emotional stress, logistical hurdles, and financial burdens associated with traveling to prisons for visits. Additionally, the current system of communication is inefficient and time-consuming, often resulting in reduced or missed opportunities for prisoners to maintain meaningful connections with their loved ones. This lack of regular family contact negatively impacts the mental wellbeing of both prisoners and their families, hindering rehabilitation efforts and reintegration into society post-incarceration.

1.2 Purpose (Of Your Solution/Software)

The Wasl System is designed to revolutionize communication between prisoners and their families by providing a secure, user-friendly platform for virtual visitations. By facilitating online communication, the system eliminates the need for physical travel, reduces emotional and financial strain, and enhances the frequency of contact. Wasl aims to promote family cohesion and emotional support while adhering to all security protocols required in a correctional environment. The system also seeks to support prison reform by enhancing the rehabilitative aspects of incarceration.

1.3 Scope (Domain and Main Focus of the Solution)

The Wasl System focuses on delivering a reliable and secure platform for virtual visitations and online communication between prisoners and their families. The main features include:

• Scheduling and conducting online visitation sessions

• Modifying personal information and preferences for users

• Customizing notifications and reminders for upcoming visits

• Ensuring secure communication, authenticated by government entities like Absher

• Implementing AI features such as face recognition

• Using AI/ML to implement smart scheduling

• Develop a dashboard and management console for administrators

The system is specifically designed for use within the correctional facilities domain, supporting communication between inmates, family members, and prison officials, all while complying with legal, regulatory, and security guidelines.

1.4 Definitions, Acronyms, and Abbreviations

• SRS: Software Requirements Specification – A document that outlines both functional and non-functional requirements of the Wasl System

• TLS: Transport Layer Security – A protocol used to secure communications over a computer network

• Absher: A Saudi government service platform used for the verification of identities and user authentication

• CI/CD: Continuous Integration/Continuous Deployment – A development practice used to automatically test and deploy changes to the system

• SSL: Secure Sockets Layer – A protocol for establishing authenticated and encrypted links between networked computers

• MFA: Multi-factor Authentication – A security system that requires multiple methods of authentication from independent categories of credentials

1.5 References (Dependencies, Policies, Regulations)

1. Vision and Scope Document Wasl System – Provides a comprehensive view of the project's goals and future releases

2. CI/CD Best Practices by Sushant Kapare – Explains industry best practices for continuous integration and deployment pipelines.

3. Data Protection and Privacy Laws Guide (World Bank) – Ensures compliance with international standards for protecting user data, privacy, and security.

4. TLS vs SSL Protocol Comparison (Elegant Themes Blog) – Highlights the importance of using TLS over SSL for secure communication in web applications.

5. Prison Policies and Security Protocols – Compliance with institutional policies, including regulations from the Ministry of Justice

2. ARCHITECTURAL REPRESENTATIONS: 4+1 VIEW MODEL OVERVIEW

2.1 Introduction to the 4+1 View Model

The 4+1 View Model is a framework designed to describe the architecture of software intensive systems. It organizes the architecture into five concurrent views, each addressing different stakeholder concerns. These views are:

• Logical View: Focuses on the functionality and structure of the system

• Development View: Shows how the system is organized from a development perspective

• Process View: Describes the dynamic behavior of the system and its runtime aspects

• Physical View: Illustrates the physical deployment of the software on hardware

• Scenarios (Use Case View): Ties together all other views by showing how the system supports critical use cases

Figure 1: 4+1 View Model Architecture Overview

3. LOGICAL VIEW

3.1 Overview

The Logical View focuses on the functional requirements of the Wasl System and illustrates how the system is structured to handle key operations. This view primarily represents the relationships between key classes, components, and subsystems within the software, defining how they interact to fulfill the system's requirements.

In the Wasl System, the Logical View includes core components such as:

• User Management: Handles the authentication, authorization, and profile management of users

• Visit Request Management: Manages the creation, approval, and scheduling of online visit requests between family members and prisoners

• Notification System: Sends notifications to users about visit approvals, reminders, and status updates

• Session Management: Manages the initiation and completion of online visitation sessions

This view provides a high-level overview of how each of these components interacts, ensuring that the system fulfills the core functionalities for both end-users and administrators.

3.2 Class Diagram

The Class Diagram for the Wasl System provides a comprehensive structural representation of the system's core entities, their attributes, methods, and interconnected relationships. It illustrates how the system components work together to facilitate secure virtual visitations between prisoners and their families.

Figure 2: Wasl System Class Diagram

3.2.1 Core Entity Classes:

• User (Abstract Base Class): Serves as the foundation for all system users, containing common attributes such as userId, name, email, phoneNumber, and isActive status. It provides essential methods like updateProfile() for managing user information across all user types.

• FamilyMember: Inherits from User and represents family members seeking to visit prisoners. Key attributes include relationship type, absherVerified status for government authentication, and user preferences. It provides specialized methods such as requestVisit() for initiating visit requests and authenticateAbsher() for secure government verification.

• Prisoner: Also inheriting from User, this class represents incarcerated individuals with specific attributes including prisonerId, cellBlock location, and visitingPrivileges status. It offers methods like acceptVisitRequest() for responding to visit invitations and getAvailableSlots() for scheduling coordination.

• Administrator: Extends the User class to represent prison officials and system administrators. Contains attributes for adminLevel, prisonFacility assignment, and permissions list. Provides administrative methods including approveVisitRequest() for visit authorization and manageDashboard() for system oversight.

3.2.2 Request and Session Management:

• VisitRequest: Central to the visit management process, this class tracks all visit-related information including requestId, familyMemberId, prisonerId, requestedDateTime, status (using RequestStatus enum), and duration. It provides methods for submitRequest() and updateStatus() to manage the visit approval workflow.

• VisitSession: Represents active virtual visit sessions with attributes such as sessionId, visitRequestId reference, startTime, endTime, sessionStatus, and recordingEnabled flag. Key methods include initiateSession() and terminateSession() for managing the actual video communication.

3.2.3 Supporting Service Classes:

• NotificationService: Handles all system communications with attributes including notificationId, userId, message content, notificationType, and sentDateTime. Provides the sendNotification() method to deliver alerts, reminders, and status updates to users.

• AIService: Implements artificial intelligence features with attributes for aiServiceId, faceRecognitionEngine, and smartScheduler components. Methods include authenticateFace() for biometric verification, suggestOptimalTime() for intelligent scheduling, and analyzeUsagePatterns() for system optimization.

• SecurityService: Ensures system security and compliance with attributes such as securityId, encryptionLevel, and absherIntegration status. Provides critical security methods including authenticateUser(), validateAbsher() for government verification, and encryptCommunication() for secure data transmission.

• Dashboard: Facilitates administrative oversight with attributes including dashboardId, userRole, and widgets list for customizable interface elements. Methods include displayStatistics() for system analytics, manageUsers() for user administration, and generateReports() for comprehensive system reporting.

3.2.4 Data Types and Enumerations:

RequestStatus: An enumeration defining the possible states of visit requests including PENDING, APPROVED, REJECTED, and COMPLETED, ensuring consistent status tracking throughout the system.

3.2.5 Relationships and Interactions:

The class diagram demonstrates clear inheritance relationships where FamilyMember, Prisoner, and Administrator classes inherit common functionality from the abstract User base class. Association relationships show how FamilyMembers create VisitRequests, Prisoners receive and respond to these requests, and VisitRequests generate VisitSessions upon approval. The NotificationService maintains associations with all User types to provide comprehensive communication capabilities, while the AIService and SecurityService support various system operations through their specialized functionalities.

This structural design ensures modularity, maintainability, and scalability while supporting all core requirements including secure communication, AI-enhanced features, government authentication integration, and comprehensive administrative management capabilities.

3.3 Collaboration Diagrams

A Collaboration Diagram shows how objects interact with one another to achieve a specific functionality or use case in the system. It highlights the relationships between objects and the sequence of messages exchanged to perform tasks. In the Wasl System, a collaboration diagram for the "Request Online Visit" use case shows the following interactions:

Figure 3: Collaboration Diagram - Request Online Visit

3.3.1 Main Interaction Flow:

1. Request Creation: The FamilyMember object interacts with the VisitRequest object to create and submit a new visit request with prisoner details and preferred timing

2. Administrator Notification: The VisitRequest object communicates with the Notification object to send alerts to prison officials about the pending request that needs approval

3. Alert Delivery: The Notification object sends the approval alert to the Administrator object, providing all necessary request details for review

4. Request Approval: The Administrator object responds to the VisitRequest object with their approval or rejection decision

5. User Notifications: Once approved, the VisitRequest object asks the Notification object to inform both the family member and prisoner about the scheduled visit

6. Family Confirmation: The Notification object sends visit confirmation and session details to the FamilyMember object

7. Prisoner Alert: The Notification object informs the Prisoner object about the upcoming scheduled visit

8. Session Creation: The approved VisitRequest object creates a VisitSession object to manage the scheduled online meeting

These diagrams provide a dynamic view of how objects cooperate within the system to complete various tasks, ensuring smooth functionality of the online visitation process.

3.4 Sequence Diagrams

A Sequence Diagram illustrates the interactions between objects in a specific sequence over time, focusing on the chronological order of messages exchanged to perform a particular operation. In the Wasl System, a sequence diagram for the "Schedule Online Visit" use case would show the following sequence of interactions:

Figure 4: Sequence Diagram - Schedule Online Visit with AI Integration

3.4.1 Main Sequence Flow:

• User Initiates Request

• The User calls createRequest() on the VisitRequest object.

• Validation

• VisitRequest verifies the request data and sends a confirmation message back to the User.

• Notification to Officials

• VisitRequest communicates with the Notification object to send alerts to the appropriate PrisonOfficials.

• AI Smart Scheduling Consultation

• VisitRequest consults the AIService to request suggested time slots.

• AIService analyzes:

• Prisoner availability

• Family preferences

• Usage patterns (e.g., peak vs. off-peak times)

• Past visitation history

• AIService returns one or more optimal time slots.

• Review by Prison Officials

• PrisonOfficials review the request (including AI-suggested slots).

• They send back either approveRequest() or rejectRequest() to the VisitRequest.

• Decision Handling

• If Rejected:

• VisitRequest notifies the User of rejection.

• If Approved:

• VisitRequest re-engages the AIService to finalize the best slot.

• AIService confirms an optimal time.

• A new Session object is created.

• Session Notification

• The Session object sends confirmation messages to both the User and the Prisoner.

The Sequence Diagram visually depicts how objects interact in the order they do, clarifying the flow of events and data within the system for specific functionalities like visit scheduling. The integration of AI services ensures intelligent scheduling optimization throughout the process.

4. PROCESS VIEW

4.1 Overview

The Process View of the Wasl System focuses on the dynamic aspects of the system, such as how it behaves at runtime and manages concurrent processes. It describes how various components interact during the execution of tasks, ensuring the system operates efficiently under different scenarios. For the Wasl System, the Process View shows how user requests (like scheduling online visits) are processed, from user input to interaction with backend services like authentication, visit management, notification systems, and AI-powered smart scheduling.

It also highlights how the system manages concurrent user sessions, processes requests in real-time, and ensures smooth operation for multiple users simultaneously. The AI component plays a crucial role in optimizing scheduling and resource allocation during peak usage periods.

This view helps stakeholders understand the system's runtime behavior, performance, scalability, and how it handles critical processes, such as visit scheduling, notification sending, session management, and AI-driven optimizations.

4.2 Activity Diagram

The Activity Diagram illustrates the workflow and decision points in the visit scheduling process, including AI integration for smart scheduling  
Figure 5: Activity Diagram - Visit Scheduling Process with AI Integration

The diagram shows the complete workflow from user login through visit confirmation, including decision points for authentication verification and approval status. When the system detects available dates and time slots, it provides options for users to select their preferred timing. If authentication fails, users are redirected to re-authenticate, and if requests are rejected, appropriate notifications are sent with reasons for denial.

5. DEVELOPMENT VIEW

5.1 Overview

The Development View of the Wasl System provides a structured overview of the system's architecture from a developer's perspective. It focuses on the organization of the software modules, components, and their interactions, facilitating a clear understanding of how the system is constructed and maintained.

In the Wasl System, the Development View highlights the following key components:

• User Interface (UI): The front-end component that allows users (family members, prisoners, and officials) to interact with the system. It includes web and mobile applications designed for ease of use and accessibility.

• Visit Management Service: A backend component responsible for handling visit requests, scheduling, and management. It processes user inputs, manages visit statuses, and interfaces with the database.

• Notification Service: This component handles all notifications related to visit requests, approvals, and reminders, ensuring that users receive timely information about their visit schedules.

• Authentication Service: Responsible for managing user login and security, verifying user identities, and ensuring secure access to the system.

• Session Management Service: This component manages the creation, execution, and termination of online visit sessions, ensuring a smooth experience for users during virtual visitations.

• AI Services: Integrated artificial intelligence components for face recognition, smart scheduling optimization, and usage pattern analysis to enhance system efficiency and security.

The Development View helps developers understand the modular structure of the system, enabling efficient development, testing, and maintenance processes. It also highlights the dependencies between components, making it easier to implement changes and enhancements while maintaining system integrity.

5.2 Component Diagram

The Component Diagram showcases the structure of the Wasl System, depicting how the User Interface interacts with various backend services, such as the Authentication, Visit Management, Notification Services, and AI Services. These services, in turn, communicate with databases that store user and visit-related data, ensuring smooth system functionality.

Figure 6: Component Diagram - System Architecture

6. PHYSICAL VIEW

6.1 Overview

The Physical View of the Wasl System illustrates the physical deployment of the software components on hardware infrastructure. It provides insights into how the system is structured in a real-world environment, detailing the distribution of software across servers and devices.

In the Wasl System, the Physical View includes the following key elements:

• User Devices: This encompasses the various platforms through which users access the Wasl System, including mobile applications (for family members) and web applications (for both family members and prison officials).

• Web Server: The web server hosts the front-end components of the Wasl System, handling user requests, serving web pages, and managing client-side interactions. It acts as the interface between the users and the application server.

• Application Server: This server is responsible for executing the business logic of the Wasl System. It processes requests related to visit management, authentication, notifications, and AI services. The application server interacts with the database to store and retrieve data.

• Database Server: The database server stores all persistent data, including user profiles, visit requests, session details, notifications, and AI processing results. It ensures data integrity and security, providing reliable access for the application server.

• Network Infrastructure: This includes the necessary networking components, such as routers and switches, that facilitate communication between user devices and servers, ensuring seamless connectivity and data transfer.

• AI Processing Infrastructure: Dedicated resources for handling face recognition, smart scheduling algorithms, and usage analytics to support the system's intelligent features.

The Physical View is crucial for understanding how the Wasl System operates in a production environment, highlighting the hardware requirements, network topology, and deployment architecture necessary for optimal performance and scalability. It helps in planning for resource allocation, ensuring that the system can handle expected user loads and maintain availability and responsiveness.

6.2 Deployment Diagram

The Deployment Diagram illustrates the physical infrastructure of the Wasl System. It shows how users interact with the system via Mobile or Web Apps, which communicate with a Web Server. The Application Server handles core logic, while the Database Server manages data storage for user information and visit requests.

Figure 7: Deployment Diagram - Physical Infrastructure

The diagram clearly shows the three-tier architecture with user devices connecting to web servers, which then communicate with application servers for business logic processing, and finally interact with database servers for data persistence.

7. QUALITY ATTRIBUTES AND SCENARIOS

7.1 Important Quality Attributes Related to the Project

• Security: The Wasl System must ensure secure communication between family members, prisoners, and prison officials, following legal requirements. It should use strong authentication mechanisms (e.g., MFA, encryption like TLS).

• Usability: The system should be user-friendly for family members, prisoners, and officials, providing a seamless interface for scheduling visits and managing notifications.

• Scalability: The system must handle increasing numbers of users (families, prisoners, officials) without degrading performance, especially during peak times for online visit requests.

• Availability: The Wasl System needs high availability to ensure that users can access it at any time, given the sensitive nature of prisoner-family communication.

7.2 Quality Attribute Scenarios

7.2.1 Scenario for Security

- Description: A family member logs into the system and requests a visit with a prisoner. The system ensures that only authorized individuals can access the request and session data, using multi-factor authentication (MFA) and encrypted communication via TLS.

- Steps:

• Family member logs into the system using username and password.

• System requests additional authentication (MFA).

• User is authenticated, and the system grants access to the scheduling page.

• Communication between user and the server is secured via TLS.

Figure 8: Security Scenario Flow Diagram

7.2.2 Scenario for Usability

Description: A user with limited technical skills accesses the Wasl System to schedule a visit. The interface guides the user through simple steps to select a time slot and confirm the visit.

- Steps:

• User navigates to the "Request Visit" page.

• User is guided step-by-step to select a date, time, and confirm the visit.

• System provides confirmation and sends notification to both user and prisoner.

Figure 9: Usability Scenario Flow Diagram

7.2.3 Scenario for Scalability

- Description: During a busy holiday period, the Wasl System receives a high volume of visit requests. The system scales up its resources to handle the increased load without performance degradation.

- Steps:

• The system automatically detects increased traffic.

• Additional server resources are allocated to handle the load.

• Users continue to schedule visits with no slowdowns.

Figure 10: Scalability Scenario Flow Diagram

7.2.4 Scenario for Availability

- Description: A user attempts to access the Wasl System for visit scheduling outside of business hours. The system is available 24/7, ensuring the user can schedule a visit at any time.

- Steps:

• Users log into the system during non-peak hours.

• The system is available and allows the user to access all functionality.

• The user schedules a visit and receives confirmation.

Figure 11: Availability Scenario Flow Diagram

8. ARCHITECTURAL STYLE: SERVICE-ORIENTED ARCHITECTURE (SOA)

8.1 Why This Style Was Chosen:

• Modularity: SOA breaks the system into independent services (e.g., User Management, Visit Request Management, Notification Service, AI Services), which can be developed and maintained independently. This modularity aligns well with the Wasl System's complex functionality.

• Scalability: SOA allows each service to scale independently based on load. For example, during high demand for visit scheduling, the Visit Request Management Service can scale up without affecting other services.

• Security: SOA supports the use of security protocols like TLS and the implementation of secure APIs, ensuring that sensitive communication (e.g., visit requests, user data) is protected.

• Interoperability: SOA is ideal for integrating with external systems like Absher for identity verification. Each service can interact with external APIs, providing flexibility.

• AI Integration: SOA facilitates the integration of AI services as independent components that can be easily updated and scaled without affecting the core system functionality.

8.2 Implementation in Wasl System:

The system will be broken down into independent services, such as:

Figure 13: Service-Oriented Architecture Diagram

• Authentication Service: Manages user logins and security, integrates with Absher for verification.

• Visit Management Service: Handles all visit requests, approvals, and scheduling.

• Notification Service: Sends notifications to users regarding their visit status.

• Session Management Service: Manages the initiation and conclusion of online visit sessions.

• AI Services: Handles face recognition, smart scheduling, and usage analytics. It also determines the emotion of a prisoner based on different factors.

Each service will communicate through secure APIs, ensuring modularity, security, and ease of scaling.

9. TECHNOLOGY STACK

Frontend Web: React.js

Frontend Mobile: React Native/Swift

Database: MongoDB

Backend: Express.js

Cloud Infrastructure: Railway (for microservices deployment)

AI/ML Services: AWS Rekognition (face recognition), custom ML algorithms (smart scheduling)

Authentication: Absher API integration, JWT tokens

Communication: WebRTC for video sessions

10. DOMAIN MODEL

10.1 Core Entities:

• Prisoner: prisonerId, name, cellNumber, status, biometricsTemplateId, allowedContacts[], visitationPreferences

• FamilyMember: familyId, name, relationship, contactInfo (email, phone), country, AbsherId (optional), verified

• VisitRequest: requestId, prisonerId, familyId, requestedSlot, status (Pending/Approved/Rejected/Completed/Cancelled), reason

• Session: sessionId, requestId, scheduledStart, scheduledEnd, actualStart, actualEnd, recordingLocation (if allowed), status

• AuthRecord: authId, userId, authType (Absher, local), mfaStatus, lastLogin

• Device: deviceId, ownerId, deviceType, lastSeen, ipInfo

• FaceBiometrics: templateId, prisonerId, enrollmentDate, matchThreshold

• Notification: notificationId, targetUserId, type, content, sentAt, status

• SlotAvailability: slotId, prisonerId, staffApproved, capacity, location (virtual/room)

• AuditLog: auditId, actorId, action, target, timestamp, metadata

• Admin: adminId, role, permissions

• UsageMetric: metricId, date, activeSessions, requestsCreated, avgWaitTime

11. USE CASES & SCENARIOS

11.1 UC-01: Request Visit (Family)

Primary Actor: FamilyMember

Goal: Book a visit with a prisoner

Main Flow (Basic Success):

• Family logs in (Absher or local + MFA).

• Family searches/selects the prisoner.

• Family views available time slots with AI-powered recommendations.

• Family submits a VisitRequest (select slot, add notes).

• System validates family eligibility and slot capacity.

• System places request in Pending and notifies prison staff.

• Staff approves the request.

• System schedules Session with AI optimization and sends notifications.

• Family and prisoner join session at scheduled time.

Alternative Flows:

• 3a. No slot available: System offers "Join Waitlist" or suggests alternate times using Smart Slot Allocation.

• 5a. Eligibility failure: System rejects and notifies family with reason.

Exception Flows:

• 2a. Authentication fails: Halt and prompt re-authentication.

• 7a. Staff rejects request: System notifies family with reason and possible appeals.

11.2 UC-02: Conduct Session (Prisoner & Family)

Primary Actors: Prisoner, FamilyMember, Staff

Main Flow:

• At scheduled time, the system opens the virtual session room.

• Prisoners log in via prison kiosk (biometrics + session token). Family logs in remotely.

• Before starting, the system runs Face Recognition on prisoner device to confirm identity.

• If recognition is OK, the session begins. Staff has monitoring controls (mute, end, chat monitoring).

• Session ends; system records metadata and stores recording if permitted.

Exceptions:

• 3a. Recognition fails: Staff is alerted; session can proceed only after manual staff confirmation.

• 4a. Network failure: Attempt reconnect; if fail, mark session Interrupted and offer reschedule.

11.3 UC-03: Enroll Prisoner Biometric

Primary Actor: Staff

Main Flow:

1. Staff opens prisoner profile, selects 'Enroll Face'.

2. Staff captures images/videos via secure kiosk; system creates FaceBiometrics template.

3. System validates template quality. If low, request re-enrollment.

4. Save template and link to prisoner profile.

Exceptions:

- Hardware error: Log and route to maintenance.

11.4 UC-04: Smart Slot Recommendation

Primary Actor: System (AI assistant)

Main Flow:

1. Family requests available times.

2. System calculates best-fit slots based on prisoner schedule, family timezone, historical acceptance rates, staff availability using AI algorithms.

3. System displays prioritized suggestions for family to pick.

Exceptions:

- Insufficient data: System shows generic slots and asks for preferences.

11.5 UC-05: Management Dashboard Reporting

Primary Actor: Admin/Management

Main Flow:

1. Admin logs into dashboard.

2. Admin selects time range, filter by prison, module (sessions/requests/usage).

3. System returns visual charts (peak usage, avg session length, failed recognitions).

4. Admin can export reports and set alerts.

12. USER MANUAL DRAFT

12.1 Family Member — How to Request a Visit

1. Sign in using Absher (or local account + MFA).

2. Search for the prisoner by name or ID.

3. Click "Request Visit" and choose a suggested slot or view all available slots.

4. Confirm details and submit.

5. You will receive email/SMS reminders; on the day, click "Join Session" 5 minutes early and ensure your camera/mic work.

12.2 Prison Staff — Enroll Biometric

1. Login to Staff Portal (2FA required).

2. Open Prisoner profile -> Enroll Face.

3. Use secure kiosk to capture images; follow quality prompts.

4. Save template and verify sample match.

12.3 Admin — Dashboard & Reports

1. Login to Admin Dashboard.

2. Use filters to view utilization metrics.

3. Export CSV or schedule automated reports.

12.4 Troubleshooting (Common)

- Face Recognition Fails: Retake enrollment image; escalate to staff for manual verification.

- Connectivity Issues: Use backup room/phone line; reschedule the session.

- Authentication Errors: Reauthenticate via Absher, check MFA device.

13. HYPOTHESIS TESTING & EXPERIMENTS

H1: Smart Slot Recommendation reduces time-to-book by 30% and reduces cancellations by 20%.

- Experiment: A/B test UI with/without smart suggestions for 8 weeks. Measure booking time, cancellations.

H2: Face recognition will correctly identify prisoner for 95% of legitimate attempts.

- Experiment: Controlled test with enrolled prisoners (n≥ 200) across different lighting conditions and angles. This font style should be the same as others.

H3: Virtual visits increase family-reported wellbeing scores after 3 months.

- Experiment: Survey families who use Wasl vs control group; use validated wellbeing questionnaire.

Each experiment should define success criteria, monitoring metrics, sample size, duration, and rollback criteria.

14. SEQUENCE DIAGRAMS

14.1 Booking → Approval → Scheduling

This diagram illustrates the comprehensive flow of a visit booking, from the initial request by a family member to its final scheduling, incorporating AI services for optimization. It details the interactions between the User, VisitRequest, Notification, AIService, PrisonOfficials, and Session objects, highlighting key steps such as request creation, validation, smart scheduling consultation, official review, and session finalization.

Figure 14: Sequence Diagram - Complete Visit Booking Process with AI Services

14.2 Session Start with Face Recognition

This diagram demonstrates the complete flow from session initiation through face recognition verification, including AI service interactions for identity confirmation and staff oversight capabilities.

Figure 15: Session Start with Face Recognition Flow

15. SYSTEM ARCHITECTURE OVERVIEW

The system architecture follows a microservices approach with the following components:

- API Gateway → entry point for web & mobile applications

- Auth (Absher/Cognito) → identity federation and authentication management

- Scheduler → visit requests & smart slot allocation with AI optimization

- Session Orchestrator (WebRTC/Chime) → manages video call sessions

- Biometric Service (Rekognition) → ensures correct prisoner identification

- Notifications (SNS) → SMS/Email/Push notification delivery

- Admin Dashboard + Monitoring → usage analytics & security logs

- VPC Core → Database, storage, and audit logs

- Prison Kiosks → secure terminals in prison networks

Figure 16: Complete System Architecture Diagram

16. UI DESIGN GUIDELINES

16.1 General Principles

- Simplicity First: Interfaces should reduce cognitive load — only show relevant options for each user role (Family, Staff, Admin).

- Consistency: Use a unified design system (colors, typography, button styles, spacing).

- Accessibility: Comply with WCAG 2.1 AA (high contrast, alt text, keyboard navigation, screen reader support).

- Security UX: MFA prompts, consent dialogs, and session start confirmations should be clear, unambiguous, and user-friendly.

- Localization & Multilingual: Arabic (RTL) and English (LTR) supported; allow seamless switching.

16.2 Visual Identity

- Color Palette:

○ Primary: Royal Blue (trust, security)

○ Secondary: Warm Green (connection, rehabilitation)

○ Accent: Orange/Yellow (alerts, reminders)

○ Error: Red (critical issues)

- Typography:

○ Sans-serif (e.g., Inter, Noto Sans Arabic for RTL support).

○ Sizes: Headings (20–24px), Body (14–16px), Labels/Buttons (12–14px).

- Iconography:

○ Use clear, universal icons (camera, calendar, lock).

○ Support bi-directional icons for RTL (mirrored arrows, back buttons).

16.3 Layout & Navigation

- Family Portal:

○ Dashboard → Upcoming Visits, Notifications, "Book a Visit" CTA.

○ Guided booking flow (wizard-style: Prisoner → Slot → Confirmation).

○ Session launch button visible 10 min before scheduled time.

- Prison Staff Portal:

○ Sidebar navigation (Prisoner Management, Requests, Biometric Enrollment).

○ Approval workflows with status filters (Pending / Approved / Rejected).

○ Clear alerts for failed recognitions.

- Admin Dashboard:

○ Analytics panels (usage graphs, failed sessions, peak times).

○ Export and drill-down capabilities.

○ Role-based access (management vs IT admin).

16.4 UI Patterns & Components

- Forms: Minimal fields, inline validation, clear error messages.

- Tables & Lists: Paginated, searchable, with role-based filters.

- Notifications: Toast messages for success, banners for warnings, modals for critical actions.

- Buttons:

○ Primary (solid blue, for main actions).

○ Secondary (outline green, for optional actions).

○ Destructive (solid red, for cancellation/rejection).

16.5 Accessibility & Inclusivity

- RTL/LTR Switching: Full mirrored layouts.

- Contrast Ratio: Minimum 4.5:1 for text.

- Captions: Optional live captioning for sessions (family accessibility).

- Multi-device Support: Responsive layouts for desktop, tablet, mobile.

17. AGILE FEATURE BREAKDOWN

Sprint 1 — Foundations & Access Control

- User Authentication (Families, Staff, Admins).

- Family Database integration

- Prisoner database integration (read-only).

- Git Setup

Deliverable: Login + Dashboard skeleton for all roles.

18. DEPLOYMENT PLAN

18.1 Deployment Environments

- Development

○ Local dev + Railway preview deployments.

○ Expo Go for mobile preview (not deployed to stores).

○ Web frontend previews via Vercel (per branch).

- Staging (UAT)

○ Railway staging instances (isolated DB + services).

○ Vercel staging frontend (connected to staging backend).

○ Mobile via Expo Go with staging API base URL.

○ Used for testing new features + stakeholder demos.

- Production

○ Railway production instances for microservices.

○ Managed DB on Railway (Postgres/MySQL + Redis for cache).

○ Vercel production deployment for web app.

○ Expo Go (local-only mobile app) — families & staff scan QR to open app (no App Store/Play Store).

18.2 Microservices Breakdown on Railway

- Auth Service: Handles Absher API integration, MFA, and JWT issuance.

- Scheduling Service: Visit booking, smart slot allocation, session approvals.

- Session Service: Manages WebRTC sessions (secure signaling, prisoner ↔ family video).

- Biometric Service: Prisoner face recognition (AWS Rekognition API calls).

- Notification Service: SMS/Email reminders (using SNS/Twilio integration).

- Admin Dashboard Service: Provides usage metrics + analytics API.

- Database Service: MongoDB + encrypted storage.

18.3 Deployment Workflow

- CI/CD Pipeline

○ Push → GitHub/GitLab.

○ CI runs tests (unit, integration).

○ On merge → Auto deploy microservice to Railway.

○ Vercel deploys the frontend automatically on push.

○ Expo Go mobile app fetches latest code via expo publish.

- Zero Downtime Strategy

○ Railway auto restarts services on update.

○ DB migrations run via Railway CLI with backward compatibility.

18.4 Mobile Strategy (Expo Go Only)

- App not published on Play Store/iOS App Store due to iOS restrictions.

- Families & staff access the mobile app via:

○ QR code (Expo Go).

○ Direct link to Expo project.

- Updates are instant via expo publish (no need for store approval).

18.5 Pre-Deployment Checklist

- Configure Railway secrets (DB URLs, Absher keys, JWT secrets).

- Provision Railway Postgres instance (prod + staging).

- Set Vercel environment variables (API base URLs).

- Test Expo Go with staging backend before production cutover.

- Load & security test Railway services.

- Media will be uploaded on S3 bucket.

18.6 Post-Deployment Steps

- Monitor logs in Railway dashboard (API errors, latency).

- Monitor Vercel analytics (frontend performance).

- Collect family/staff pilot feedback.

- Setup alerting (Railway health checks + email alerts).