**Project Report**

**Boston Aid and Dispatch Assistant System (BAD-AS)**

**1. Abstract**

The Boston Aid and Dispatch Assistant System (BAD-AS) is a cutting-edge emergency response and healthcare management platform designed to optimize ambulance dispatch, patient tracking, and hospital coordination. This system incorporates role-based access control to accommodate multiple users, including doctors, nurses, emergency coordinators, and IT support personnel. Developed using Java Swing, BAD-AS facilitates real-time symptom analysis, efficient patient record management, and data-driven ambulance recommendations. This report provides an in-depth analysis of the system’s architecture, design principles, security measures, and performance considerations, along with potential future enhancements.

**2. Introduction**

**2.1 Project Background**

Efficient coordination between hospitals, emergency dispatchers, and medical professionals is critical for minimizing delays in patient care. BAD-AS is designed to bridge communication gaps by offering a comprehensive, integrated platform that streamlines emergency case management, ambulance dispatch, and symptom analysis.

**2.2 Objectives**

The primary goals of the BAD-AS system include:

* Implementing secure, role-based authentication to regulate access.
* Enabling real-time symptom assessment to assist in prioritizing emergency cases.
* Developing an ambulance recommendation feature based on patient severity.
* Streamlining hospital and medical staff management.
* Ensuring data security and integrity for all stored patient and hospital records.

**2.3 Scope**

BAD-AS is designed to manage:

* Role-specific authentication for administrators, doctors, nurses, dispatchers, and IT support personnel.
* Patient data management via an intuitive GUI.
* Emergency dispatch coordination and ambulance allocation.
* Secure storage of patient, hospital, and dispatch records, with potential migration from CSV files to a more robust database system.

**3. System Overview**

**3.1 Architecture & Design**

The system follows a modular design, ensuring that each user role has a dedicated interface tailored to their specific responsibilities. Developed using Java Swing for the frontend and CSV files for temporary storage, the system has been designed with future database integration in mind.

**3.1.1 High-Level Architecture**

BAD-AS consists of the following components:

* **Frontend:** A Java Swing-based graphical user interface (GUI) for user interaction.
* **Backend:** Core business logic implemented in Java.
* **Storage Layer:** CSV files for temporary data storage, with potential transition to database integration.

**3.2 Functional Modules**

**3.2.1 User Authentication**

* Implemented in Login.java
* Manages credential storage in CSV files
* Supports role-based access control

**3.2.2 Patient Management**

* **ReceptionistPage.java:** Handles patient registration.
* **DoctorPage.java:** Enables doctors to diagnose patients and prescribe treatments.
* **NursePage.java:** Allows nurses to update patient records.

**3.2.3 Emergency Coordination**

* **DispatcherPage.java:** Assigns ambulances to emergency requests.
* **AmbulanceRecommendation.java:** Uses symptom analysis for prioritizing dispatch.

**3.2.4 Hospital & IT Support**

* **AdminPage.java:** Manages hospitals and medical staff.
* **ITSupportPage.java:** Handles user authentication and security management.

**3.3 Data Flow & Storage**

BAD-AS utilizes CSV files for data storage, including:

* patient\_records.csv – Stores patient information, diagnoses, and prescriptions.
* dispatch\_requests.csv – Logs ambulance dispatch details.
* hospitals.csv – Maintains hospital records and details.

**4. Implementation Details**

**4.1 Technologies Used**

* **Java Swing** – Frontend UI development and business logic implementation.
* **CSV Files** – Temporary data storage solution.
* **Event-Driven Programming** – Enables interactive UI elements through button actions.

**4.2 Code Structure & Key Classes**

|  |  |
| --- | --- |
| Class Name | Functionality |
| Login.java | Manages user authentication |
| ReceptionistPage.java | Handles patient registration |
| DoctorPage.java | Enables diagnosis and prescription management |
| NursePage.java | Updates patient treatment records |
| DispatcherPage.java | Manages ambulance dispatch requests |
| AmbulanceRecommendation.java | Assists in emergency prioritization |
| AdminPage.java | Oversees hospital and staff management |
| ITSupportPage.java | Maintains user authentication and security |

**4.3 User Roles & Access Control**

The system implements role-based access control:

* **Admin:** Oversees hospital and user account management.
* **Doctor:** Diagnoses patients and updates medical records.
* **Nurse:** Manages patient treatment updates.
* **Dispatcher:** Allocates ambulances based on emergency severity.
* **IT Support:** Handles authentication and access control.

**5. Security & Performance Considerations**

**5.1 Authentication & Data Security**

* **Current Issue:** Passwords are stored in plain text within CSV files, posing a security risk.
* **Proposed Solution:** Implement password hashing (e.g., BCrypt) and transition to database storage for enhanced security.

**5.2 Scalability & Optimization**

* **Current Issue:** CSV-based storage is not scalable for handling large datasets.
* **Proposed Solution:** Migrate to MySQL/PostgreSQL for improved performance.
* **Current Issue:** Swing UI experiences lag when processing large datasets.
* **Proposed Solution:** Implement multi-threading to enhance UI responsiveness.

**6. Testing & Validation**

**6.1 Test Cases**

|  |  |  |
| --- | --- | --- |
| Test Case | Expected Outcome | Status |
| User login with valid credentials | Access granted | ✅ Pass |
| User login with incorrect credentials | Access denied | ✅ Pass |
| Patient registration | Successful record storage | ✅ Pass |
| Doctor updates patient record | Record updated in patient\_records.csv | ✅ Pass |
| Dispatcher assigns an ambulance | Dispatch recorded in dispatch\_requests.csv | ✅ Pass |

**6.2 Bug Fixes & Improvements**

* Resolved UI inconsistencies in symptom analysis.
* Strengthened role-based access control mechanisms.
* Enhanced error handling to prevent application crashes.

**7. Future Enhancements**

1. **Database Migration** – Transition from CSV storage to MySQL/PostgreSQL.
2. **API Integration** – Incorporate Google Maps API for real-time ambulance tracking.
3. **Security Enhancements** – Implement encryption and password hashing.
4. **Multi-threading** – Improve UI performance and responsiveness.
5. **Mobile Application** – Develop an Android/iOS version for on-the-go access.

**8. Conclusion**

The BAD-AS system successfully implements a robust emergency response platform with role-based authentication, real-time patient management, and ambulance dispatch coordination. While the current implementation provides a strong foundation, further enhancements in security, scalability, and user experience—such as database integration and mobile application development—can significantly improve its overall efficiency and effectiveness.

**9. References**

* IEEE Standards for Software Documentation.
* Java Swing Official Documentation.
* Best Practices for Secure Authentication in Java.
* Scalability Strategies for Emergency Response Systems.