yazı tipi, grafik, grafik tasarım, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

**SENG 244 - Object Oriented Software Engineering**

**NGO Aid Operations Management System: NGO-AOMSYS**

**SENG 244- CEVAM5**

**Ahmet Bartu Bağcı- 210201017**

**Volkan Çevik- 200201031**

**Ergin Şen- 200204046**

**Mehmet Emin Çetinkaya- 210201014**

**Furkan Ceyhun Kabasakal- 200201030**

**1. Introduction**

The purpose of this Software Design Document (SDD) is to provide a comprehensive overview of the design for the NGO-AOMSYS application. This document details the system architecture, design goals, and key components. It also serves as a reference for developers and other stakeholders involved in the project.

This SDD is based on the requirements analysis document prepared for the NGO-AOMSYS project. The document will reference relevant sections of the requirements document throughout to ensure traceability between design decisions and project goals.

**1.1 System Purpose**

NGO-AOMSYS is a software system designed to meet the needs of National Non-Governmental Organizations (NGOs) that face the burden of managing the allocation of aid provisions to demand coming from geographically dispersed households that are in need. The system aims to ensure the effective allocation and distribution of in-kind or cash aid.

**1.2 Design Goals**

* Effectively manage aid requests.
* Ensure efficient resource allocation.
* Organize volunteer work.
* Provide appropriate interfaces for different user types (donors, volunteers, beneficiaries).
* Track inventory and generate reports.

**1.3 Definitions, Acronyms, and References**

* **NGO:** Non-Governmental Organization
* **AOMSYS:** Aid Organization Management System
* **GIS:** Geographic Information System

**1.4 Overview**

NGO-AOMSYS is designed as a web-based software application. The user interface is customized for different user types, including donors, volunteers, beneficiaries, and system administrators. The system provides an environment where users can register, manage their profiles, and perform system functions.

NGO-AOMSYS consists of the following core components:

* User Management: Manages user registrations, roles, and permissions.
* Donation Management: Records, tracks, and reports donations.
* Volunteer Management: Manages volunteer registrations, profiles, and assignments.
* Beneficiary Management: Records, evaluates, and tracks aid requests.
* Operation Management: Plans, tracks, and reports aid operations.
* Inventory Management: Tracks aid supplies stored in the warehouse.
* Reporting: Generates statistical reports on system usage. These components work together to ensure the efficient management of the aid process.

**2. Current Landscape of NGO Management Systems**

Since NGO-AOMSYS is a new system designed to address the needs of NGOs, there isn't a specific system being replaced. However, it's valuable to acknowledge the current landscape of NGO management systems to understand the context and potential improvements NGO-AOMSYS aims to deliver.

**Common Challenges Faced by NGOs:**

* **Manual Processes:** Many NGOs rely on manual processes for tasks like managing donor information, tracking aid requests, and scheduling volunteer activities. This can be time-consuming and error-prone.
* **Data Silos:** Information about donors, volunteers, and beneficiaries may be scattered across spreadsheets, emails, and paper documents, making it difficult to get a holistic view of operations.
* **Limited Reporting Capabilities:** Traditional methods often lack robust reporting features, making it challenging to track aid distribution, analyze resource allocation, and measure the impact of programs.

**Existing Solutions:**

Several software solutions cater to NGO management, each with varying functionalities and target audiences. Here's a brief overview:

* **Generic Project Management Tools:** Project management software like Asana or Trello can be used to manage basic tasks and communication within an NGO. However, these tools may not have features specifically tailored for aid distribution and volunteer coordination.
* **Donor Management Systems (DMS):** These systems focus on donor relationship management, allowing NGOs to track donations, send communication materials, and manage fundraising campaigns. Examples include DonorPerfect or Blackbaud Raiser's Edge NXT.
* **Volunteer Management Systems (VMS):** VMS solutions help recruit, track, and schedule volunteers. They may also offer features for volunteer training and communication. Examples include SignUpGenius or VolunteeringMatch.

**Gaps Addressed by NGO-AOMSYS:**

NGO-AOMSYS aims to bridge the gap between existing solutions by offering a comprehensive platform that integrates functionalities for:

* **Donor Management:** Donation tracking, project selection, and historical records.
* **Volunteer Management:** Volunteer registration, profile building, scheduling, and availability management.
* **Beneficiary Management:** Registration, needs assessment, and aid request tracking.
* **Operation Management:** Scheduling aid distribution, logistics optimization using algorithms, and (optionally) utilizing GIS for visualization.
* **Reporting and Analytics:** Generating reports on donations, aid distribution, volunteer activities, and program impact.

By consolidating these functionalities, NGO-AOMSYS aims to streamline NGO operations, improve data management, and enhance decision-making for a more efficient and impactful aid distribution process.

**3. Proposed Software Architecture for NGO Aid Operations Management System (NGO-AOMSYS)**

**3.1 Overview**

NGO-AOMSYS is a sophisticated software platform tailored specifically for non-governmental organizations (NGOs) to streamline and optimize their aid operations. It provides a comprehensive view of the software architecture, delineating the functionalities allocated to each subsystem for seamless coordination and efficient management.

**3.2 Subsystem Decomposition**

The subsystems within NGO-AOMSYS are designed to cater to the diverse needs of NGOs:

* **User Management Subsystem:** This subsystem encompasses user registration, role assignment, and access control. It ensures that only authorized personnel can access sensitive data and perform crucial operations within the system. Additionally, it includes features for managing user permissions dynamically based on changing roles and responsibilities.
* **Project Management Subsystem:** Here, NGOs can create, track, and evaluate aid projects from inception to completion. It includes features like project planning, budget management, progress tracking, and impact assessment. Moreover, it facilitates collaboration among team members, allowing for real-time updates and communication.
* **Inventory Management Subsystem:** Essential for tracking aid resources such as food, medical supplies, and equipment. It manages inventory levels, procurement processes, warehouse management, and distribution logistics. Furthermore, it includes functionalities for inventory forecasting, demand planning, and supplier relationship management.
* **Reporting Subsystem:** Provides comprehensive reporting and analytics capabilities. NGOs can generate various reports, including financial statements, project progress reports, donor reports, and impact assessment reports, aiding in decision-making, transparency, and accountability. Moreover, it supports customizable dashboards and data visualization tools for actionable insights.

**3.3 Hardware/Software Mapping**

NGO-AOMSYS leverages a distributed architecture to ensure scalability, reliability, and performance optimization:

* **Server Node:** Houses the central database (e.g., MySQL, PostgreSQL) and core business logic. It manages data storage, retrieval, and processing tasks, ensuring data integrity, consistency, and availability.
* **Client Nodes:** Interfaces for end-users, including desktop applications for administrators and web/mobile applications for field workers. These clients interact with the server node via secure APIs, ensuring data security, confidentiality, and seamless user experience across devices.
* **Off-the-Shelf Components:** Integration of third-party tools for enhanced functionality, such as data encryption libraries (e.g., OpenSSL), reporting frameworks (e.g., JasperReports), and authentication protocols (e.g., OAuth 2.0). This integration reduces development time, enhances system capabilities, and ensures compliance with industry standards.

**3.4 Persistent Data Management**

NGO-AOMSYS implements robust data management strategies to handle various types of persistent data:

* **Project Data:** Details of aid projects, including objectives, timelines, budgets, milestones, and beneficiary information. It supports version control, audit trails, and data archiving for historical reference and regulatory compliance.
* **Inventory Data:** Records of aid resources, including stock levels, procurement history, donations received, and distribution logs. It includes features for inventory reconciliation, quality control, and expiration management to prevent wastage and optimize resource utilization.
* **User Data:** Secure storage of user profiles, credentials, roles, and permissions. It enforces data encryption, access controls, and data masking techniques to protect sensitive information from unauthorized access or data breaches.

Data management strategies include data normalization, indexing for quick retrieval, backup and recovery plans, data retention policies, and disaster recovery mechanisms to ensure data availability, confidentiality, and integrity.

**3.5 Access Control and Security**

Security is paramount in NGO-AOMSYS, with comprehensive access control and security measures:

* **Access Control Model:** Based on the principle of least privilege, ensuring users have access only to resources necessary for their roles. It supports role-based access control (RBAC), attribute-based access control (ABAC), and dynamic access control policies for granular access management.
* **Authentication Mechanisms:** Utilizes strong authentication methods like two-factor authentication (2FA), biometric authentication, and multi-factor authentication (MFA) for enhanced security. It includes features for session management, password policies, and account lockout mechanisms to prevent unauthorized access attempts.
* **Encryption Standards:** Implements AES (Advanced Encryption Standard) for data-at-rest encryption and TLS (Transport Layer Security) for data transmission security. It supports encryption key management, cryptographic algorithms, and digital signatures to protect data confidentiality, integrity, and authenticity.
* **Security Audits and Monitoring:** Conducts regular security audits, vulnerability assessments, and penetration testing to identify and mitigate security risks. It includes intrusion detection systems (IDS), security information and event management (SIEM) tools, and real-time monitoring for proactive threat detection and incident response.
* **Compliance and Governance:** Ensures compliance with regulatory requirements (e.g., GDPR, HIPAA, ISO standards) and industry best practices for data protection, privacy, and security. It includes features for data anonymization, data masking, data minimization, and consent management to uphold ethical standards and legal obligations.

**3.6 Boundary Conditions**

NGO-AOMSYS maintains robust boundary conditions to ensure system stability, resilience, and continuity:

* **Startup Procedures:** Initialization of system components, database connections, and service dependencies during system startup. It includes health checks, configuration validation, and self-recovery mechanisms to address startup failures and ensure seamless operation.
* **Shutdown Procedures:** Graceful shutdown procedures to safely terminate system services, release resources, and preserve data integrity. It includes data backup, cache flushing, and shutdown notifications to prevent data loss or corruption during shutdown events.
* **Error Handling:** Comprehensive error handling mechanisms, exception management, and error recovery strategies to address system errors, faults, and failures. It includes error logging, error codes, error messages, and error escalation procedures for efficient troubleshooting and resolution.
* **Performance Monitoring:** Monitors system performance metrics, resource utilization, and service level agreements (SLAs) to ensure optimal performance, scalability, and responsiveness. It includes performance tuning, capacity planning, and workload balancing to optimize system resources and meet user expectations.

**3.7 Subsystem Services Glossary**

The Subsystem Services Glossary serves as a reference for defining the services provided by each subsystem, including their operations, inputs, outputs, and interfaces. It outlines the functional boundaries between subsystems, guiding the Object Design Document (ODD) where subsystem interfaces are formally specified and documented.

**4. Object Design**

**4.1 Object Design Trade-offs**

In developing the NGO Aid Operations Management System, various trade-offs were considered to ensure its effectiveness and efficiency:

* Buy vs. Build: The decision to build the system in-house was made to tailor it precisely to the organization's unique needs and workflows, ensuring maximum flexibility and scalability.
* Memory Space vs. Response Time: Emphasis was placed on optimizing response time while managing memory space efficiently to provide a seamless user experience without sacrificing system performance.
* Flexibility vs. Performance: Striking a balance between system flexibility to accommodate diverse aid initiatives and optimal performance to ensure timely operations and resource utilization.
* Security vs. Usability: Prioritizing robust security measures to safeguard sensitive donor and recipient data while maintaining a user-friendly interface to encourage engagement and participation.

**4.2 Interface Documentation Guidelines**

this section details the data structures and elements used within the NGO-AOMSYS system:

**Classes:**

* Donor
* Volunteer
* Beneficiary
* Donation
* AidRequest
* Operation
* WarehouseItem
* User (abstract)
* Administrator (inherits from User)
* OperationCoordinator (inherits from User)

**Data Attributes:**

* **String:** Used for textual data like usernames, names, addresses, project names, regions, and descriptions.
* **Integer:** Used for numerical data like user IDs, donation IDs, quantity of warehouse items, number of people, income, and monthly expenses.
* **Date:** Used to store dates like donation dates, request dates, operation dates, and registration dates.
* **Boolean:** Used for true/false values like self-delivery preference, transportation availability, and approval status.
* **Enumeration:** (Optional) Can be used to represent pre-defined categories like donation type (cash, food, clothing), aid type (furniture, cash support, medication), operation type (collection, distribution, publicity), and volunteer availability days.

**Relationships:**

Relationships between classes are not explicitly data attributes, but rather connections between them. These can be:

* **One-to-One (1:1):** A single instance of one class is associated with a single instance of another class (e.g., an Administrator inherits from a single User).
* **One-to-Many (1:N):** A single instance of one class is associated with multiple instances of another class (e.g., a Donor can make many Donations).
* **Many-to-Many (M:N):** Multiple instances of one class can be associated with multiple instances of another class (e.g., an Operation can involve many Volunteers and distribute many Donations).

**Exceptions:**

Error conditions will be handled through exceptions thrown by methods, providing informative messages about the encountered issue. Exceptions should not be used as return values to indicate success or failure.

**Collections:**

Collections of objects (e.g., list of donations in an operation) will utilize methods like getDonations() to retrieve the elements. These methods should return robust iterators that are not affected by modifications to the underlying collection during iteration.

**4.3 Packages**

The system architecture was organized into distinct packages to facilitate modularity and maintainability:

* Donor Management Package: Handles all functionalities related to donor registration, authentication, donation processing, and history tracking.
* Volunteer Management Package: Manages volunteer registration, profile creation, approval workflow, and coordination of engagement opportunities.
* Poor Assistance Application Package: Facilitates the seamless application process for individuals seeking assistance, ensuring comprehensive data collection and eligibility assessment.
* Operation Coordination Package: Coordinates aid operation planning, fundraising initiatives, and optimization algorithms to maximize impact and resource utilization.
* Admin Management Package: Empowers administrators with comprehensive control over system operations, user management, analytics, and optional GIS integration.

**4.4 Class Interfaces Glossary**

metin, el yazısı, ekran görüntüsü, Post-it notu içeren bir resim

Açıklama otomatik olarak oluşturuldu