HOPE HARVEST

Layered Architecure

B2-Group 04

2005098
2005101
2005102
2005104
2005117
2005118

BUET | 2024

KEY LAYERS

1. Presentation Layer (User Interface and API Gateway)
2. Business Logic Layer (Core Services)
3. Persistence Layer (Data Access)
4. Database Layer (Data Storage)
5. Integration Layer (External Systems and Services)
6. Infrastructure Layer (Deployment, Scalability, and Operational Support)

PRESENTATION LAYER

Purpose :

This layer is responsible for interacting with the user and handling input/output operations. It communicates with the business logic layer to perform actions and display results to the user.

Components/Services:

User Interface

- **NormalUser Interface:** Provides a responsive UI for donors to view events, make donations, track donation history, interact with chatbot and submit or check fund requests.

- **Admin Dashboard:** A control panel for managing events, overseeing donations, assigning resources, approving fund requests, and generating reports.

- **Volunteer Portal:** Enables volunteers to view and update task progress, manage availability, and collaborate with team leaders.

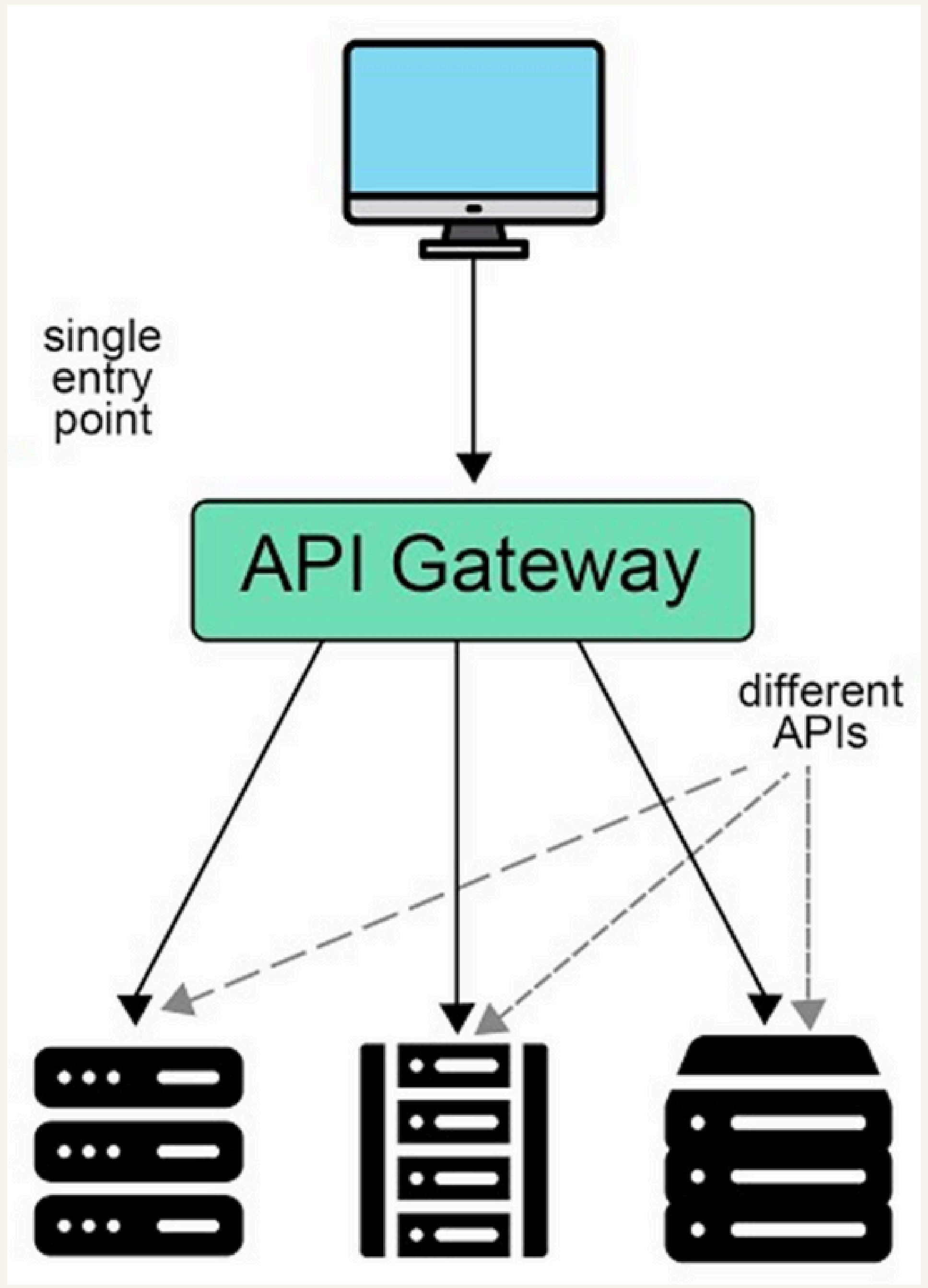
- **TeamLeader Portal**: Allows team leaders to assign tasks, manage team resources, track event progress, and submit reports.

▶ **Technology :** React.js (Component-Based Architecture, promotes reusability and modular)

API Gateway

A single entry point that routes requests from frontends to the appropriate backend services.

- **Technology :** NGINX with REST APIs (A high-performance reverse proxy and load balancer that routes client requests to REST API endpoints, ensuring secure, efficient, and modular backend communication)



BUSINESS LOGIC LAYER

Purpose :

Implements the core rules and workflows of the charity foundation system, ensuring the system's logic is independent of UI or data storage.

Key Services:

NormalUser :

- **DonationService**: Manages donation processing, tracking, and receipt generation.
- **FundRequestService:** Allows users to submit and check the status of fund requests.
- **ChatbotService:** Processes user queries and provides information via the chatbot.

Team Leaders and Volunteers :

- **TeamService**: Enables leaders to manage their teams, assign tasks, and monitor team progress.
- **ResourceService**:Allows leaders to request additional resources or update existing resource statuses.
- **EventService**: Facilitates progress tracking and submission of event updates or reports

Admins:

- **EventService:** Handles the creation, scheduling, updates, goal tracking, and status changes of events.
- **ResourceService:** Manages the allocation, updating, and tracking of resources for events and teams.
- **FundRequestService:** Enables admins to review, approve, or reject fund requests.
- **TeamService:** Manages team creation, leader assignment, and member management.
- **Centralized Accounting System Service:**
  Handles all financial workflows, including:
    - Budget allocation for events and teams.
    - Donation tracking and summaries.
    - Generating financial reports and audits.

▶ Technology :

Backend Services: Spring Boot with embedded Tomcat. (Wide Industry Adoption, Microservices-Friendly, Embedded Server Support)
RabbitMQ: To handle asynchronous updates like donation confirmations, fund disbursements, and expense tracking.

PERSISTENCE LAYER

Purpose :

Manages data storage and retrieval from databases or other storage systems. It handles CRUD (Create, Read, Update, Delete) operations and abstracts interactions with the data source.

Components/Services :

Data Access Objects(DAO)

A set of objects responsible for handling CRUD operations and complex queries for various entities.

- **Events**: EventInfo, EventReport, AuditDetails.
- **Donations**: DonationRecord, DonorInfo, PaymentStatus.
- **Users**: UserProfile (NormalUser, Admin, Volunteer), FundRequestInfo.
- **Teams**: TeamInfo, TaskAssignment, ResourceAllocation.
- **CentralAccountingSystem** : BudgetManagement, ExpenseTracking, DonationSummary, AnnualReports.

▶ **Technology :** Query Builder: Hibernate ORM (Java) – Provides object-oriented data access and supports easy migration between databases.

DATABASE LAYER

Purpose :

Provides the actual data storage for the application, ensuring that data can be queried and manipulated as needed.

Components :

Primary DB
Stores structured data such as event details, donation records, user profiles, team information, fund requests, and financial data from the centralized accounting system.
Technology: MySQL (Open-source, reliable, and compatible with enterprise tools).

Read Replicas:
Read-only copies of the primary MySQL database instance used for scaling read-heavy endpoints (e.g., event details, donation summaries). Multiple read replicas will be deployed behind a load balancer to improve performance during peak usage.
Technology: MySQL (consistent with primary storage).

INTEGRATION LAYER

Purpose:
Handles communication with external services and systems. This layer facilitates seamless integration with third-party tools, ensuring scalability and efficiency for critical processes such as payments, notifications, and data verification.

Components :

- **Payment Gateway Integration:** Connects with external payment providers (e.g., SSLCommerz, Sonali Bank, Bkash) to handle donation payments and fund disbursements.
- **Messaging and Notification Services:** Integrates with SMS and email providers to send donation confirmations, event reminders, and fund request updates.
- **Verification Services:** Integrates with external systems for user or document verification, such as ID validation or compliance checks.

Technology:

- RESTful APIs: Spring Boot for creating and consuming APIs.
- Message Queues: RabbitMQ or Kafka for asynchronous communication with external systems.

INFRASTRUCTURE LAYER

Purpose :

Provides the foundational elements for deploying, scaling, monitoring, and maintaining the system. Ensures smooth operations during peak loads, such as high donation volumes or event deadlines.

Components:

- **Hosting Platform**: Hosts the application on a scalable cloud infrastructure.
  Technology: AWS (Amazon Web Services), Google Cloud Platform (GCP), or Microsoft Azure.
- **Load Balancer:** Distributes incoming traffic among multiple server instances to ensure high availability and performance.
  Technology: Cloud-based load balancer (e.g., AWS Elastic Load Balancer).
- **Containerization and Orchestration**: Uses containers to package applications and orchestration tools for managing deployments.
  Technology: Docker for containerization, Kubernetes for orchestration.
- **CI/CD Pipeline:** Automates testing, integration, and deployment of new features and fixes.
  Technology: Jenkins, GitHub Actions, or GitLab CI/CD.
- **Monitoring and Logging:** Monitors application performance and captures logs for troubleshooting and proactive scaling.
  Technology: Prometheus and Grafana for monitoring, ELK Stack for logging.

THANK YOU