****

CALL DATA RECORD

The purpose of this document is to provide with a template for documenting Call Data Record.

**Name - Emp. ID**

Prabhath Kattupalli – 46364422

Kalandar – 46364420

Ragolu Sai Vamsi – 46364753

N. Abhilash – 46364564

Srikanth Uppari – 46364425

**1. INTRODUCTION…………………………………………………………………………4**

1.1 INTENDED AUDIENCE…………………………………………………………...4

1.2. ACRONYMS/ABBREVIATIONS…………………………………………………4

1.3. PROJECT PURPOSE……………………………………………………………….4

1.4. KEY PROJECT OBJECTIVES……………………………………………………..6

1.5. PROJECT SCOPE AND LIMITATION……………………………………………6

1.5.1. In Scope…………………………………………………………………..7

1.6. FUNCTIONAL OVERVIEW………………………………………………………7

1.7. ASSUMPTIONS, DEPENDENCIES………………………………………………8

1.8. RISKS………………………………………………………………………………8

**2. DESIGN OVERVIEW…………………………………………………………………….9**

2.1. DESIGN OBJECTIVES…………………………………………………………….9

2.1.1. Recommended Architecture………………………………….....9

2.2. ARCHITECTURAL STRATEGIES………………………………………………10

2.2.1. Creation of New Common Services/Utilities……………...10

2.2.2. User Interface Paradigms………………………………………11

2.2.3. Error Detection / Exceptional Handling………………….11

2.2.4. Concurrency and Synchronization...………………………12

**3. SYSTEM ARCHITECTURE……………………………………………………………12**

3.1. SYSTEM ARCHITECTURE DIAGRAM………………………………………...12

3.2. SYSTEM USE-CASES……………………………………………………………14

3.3. SUBSYSTEM ARCHITECTURE………………………………………………...14

3.4. SYSTEM INTERFACES………………………………………………………….15

3.4.1. Internal Interfaces………………………………………………15

3.4.2. External Interfaces……………………………………………...15

**4. DETAILED SYSTEM DESIGN…………………………………………………………15**

4.1. KEY ENTITIES…………………………………………………………………...15

4.2. DETAILED-LEVEL DATABASE DESIGN……………………………………..15

4.2.1. Data Mapping Information……………………………………...17

4.2.2. Data Conversion……………………………………………………17

4.3. DISASTER AND FAILURE RECOVERY……………………………………….17

4.4. BUSINESS PROCESS WORKFLOW……………………………………………18

4.5. BUSINESS PROCESS MODELING AND MANAGEMENT……………………18

4.6. BUSINESS LOGIC………………………………………………………………..18

4.7. VARIABLES……………………………………………………………………...18

4.8. ACTIVITY / CLASS DIAGRAMS………………………………………………..19

4.9. DATA MIGRATION……………………………………………………………...20

4.9.1. Architectural RepresentatioN………………………………20

4.9.2. Architectural Goals and Constraints……………………20

4.9.3. Logical View………………………………………………………….21

4.9.4. Architecturally Significant Design Packages……...…21

4.9.5. Data model…………………………………………………………...21

4.9.6. Deployment View…………………………………………………...21

**5. ENVIRONMENT DESCRIPTION……………………………………………………...22**

5.1. LANGUAGE SUPPORT………………………………………………………….23

5.2. USER DESKTOP REQUIREMENTS…………………………………………….23

5.3. SERVER-SIDE REQUIREMENTS……………………………………………….23

5.3.1. Deployment Considerations…………………………………...23

5.3.2 Integration Requirements………………………………………24

5.3.3. Operating System…………………………………………………..24

5.3.4. Desktop………………………………………………………………...24

**6. REFERENCES…………………………………………………………………………...24**

**7. APPENDIX………………………………………………………………………………..25**

1. **INTRODUCTION**

In the telecommunications industry, call data records, or CDRs, are essential because they document and capture critical information from each communication session. These files serve as a comprehensive log of all phone calls or data transfers that participants in a network make and receive. Call detail records (CDRs) offer crucial details such as call duration, timestamps, caller and callee names, and other contextual information. In addition to improving network optimization and correct billing, these details also significantly increase operational effectiveness and regulatory compliance.

**1.1 INTENDED AUDIENCE**

|  |  |
| --- | --- |
| CALL DATA RECORD | OPERATORS |

**1.2 ACRONYMS/ABBREVIATIONS**

|  |  |
| --- | --- |
| (CDR) | Call Data Record |
| (CB) | Customer Billing |
| (OB) | Operator Billing |

**1.3 PROJECT PURPOSE**

The Call Data Record (CDR) project serves as a multifaceted tool for telecom operators, encompassing critical functions such as billing accuracy, usage analysis, and fraud detection. By meticulously capturing and analyzing detailed records of calls, messages, and data sessions, CDRs enable operators to precisely bill customers based on their actual usage, ensuring transparency and compliance with regulatory standards. Moreover, CDR analysis unveils invaluable insights into customer behavior and service preferences, facilitating targeted marketing strategies and informed service enhancements. Beyond billing and analysis, CDRs play a pivotal role in safeguarding networks against fraudulent activities through anomaly detection and preventive measures.

**Billing Management**

**Objective:** To accurately bill subscribers based on their usage of various services provided by the mobile network.

**Details:**

• **Usage Tracking:** Capture detailed records of each subscriber's interactions such as voice calls, SMS messages, and data usage (download and upload).

• **Service Categorization:** Categorize each usage event by type (e.g., MOC, MTC, SMS-MO, SMS-MT, GPRS) to differentiate between different service types.

• **Accurate Calculation:** Calculate the duration of voice calls, count of SMS messages, and amount of data transferred accurately to reflect the exact usage of each subscriber.

• **Customer Data Presentation:** Aggregate and present usage data in a structured format for each subscriber, facilitating transparent and clear billing.

**Inter-Operator Settlement**

**Objective:** To facilitate fair and accurate settlement between different mobile operators for inter-network communications.

**Details:**

• **Cross-Network Interactions:** Capture interactions between subscribers of different mobile operators, including voice calls, SMS messages, and data usage.

• **Usage Aggregation:** Aggregate usage data across all CDR records involving inter-operator interactions.

• **Financial Reconciliation:** Provide clear and detailed usage summaries to support financial settlements between operators, ensuring fair revenue sharing and cost allocation.

• **Operator Data Presentation:** Present aggregated usage data for each operator involved in inter-network communications, supporting transparent and efficient settlement processes.

**Overall Project Goals:**

• **Accuracy:** Ensure the accuracy of billing and settlement processes by capturing and processing detailed CDR records.

• **Efficiency:** Streamline the handling and processing of large CDR files, e, to support timely billing and settlement operations.

• **Transparency:** Provide clear and structured reports for both subscribers and operators, enhancing transparency and trust within the telecom ecosystem.

**1.4. KEY PROJECT OBJECTIVES**

The primary objectives of the Call Data Record (CDR) project are as follows:

* **Client-Server System:** The system consists of a client application and a server application. The client application interacts with the server application to process user requests.
* **User Management:**  Users can sign up for the service and login to the system.
* **Customer Data Management:** The system stores and retrieves customer data.
* **Billing Information Management:** The system processes customer billing information and generates reports (CB.txt and IOSB.txt).
* **CDR File Processing:** The system processes CDR (Call Detail Record) files.

**1.5. PROJECT SCOPE AND LIMITATION**

**Functionalities included in the scope:**

* Users can sign up for the system [Store User Info in DB].
* Users can login to the system [Validate Credentials].
* The system can process customer billing requests [Customer Billing: Search MSISDN, Generate CB.txt].
* The system can process inter operator settlement billing requests [Inter operator Settlement Billing: Fetch, Generate IOSB.txt].

**The limitations of the project, as depicted by the context diagram, are:**

* The system doesn't handle credit card information directly [Client does not send credit card information to Server].
* It is not possible to know what specific methods are used for connection establishment or what data is exchanged during this process [Connection Establishment request, Connection acceptance].
* The details of how User choices are processed are not included [Process requests].
* The context diagram doesn't show how the system interacts with the bank or financial institutions.

**1.5.1. In Scope**

**The Call Data Record (CDR) project encompasses the following in-scope elements:**

**1. Data Collection and Integration**:

* Gathering CDRs from telecommunications systems, including both voice and data communications.
* Integrating CDR data from various network elements such as switches, routers, and gateways.

1. **Data Processing and Analysis**:

* Analysing CDRs to extract key call details such as timestamps, caller/callee identifiers (phone numbers or subscriber IDs), call duration, and call type (e.g., local, international, roaming).
* Processing CDR data to derive insights into network performance metrics, call traffic patterns, and usage trends.

**1.6. FUNCTIONAL OVERVIEW**

The functional overview of the Call Data Record (CDR) project encompasses the following key components:

1. **Data Collection and Integration**:

* **Purpose**: Gather CDRs from telecommunications systems and network elements.
* **Activities**: Capture call details including timestamps, caller/callee identifiers, call duration, and call types (e.g., local, international).
* **Tools**: Utilize data collection tools integrated with network switches, routers, and gateways.

1. **Data Processing and Analysis**:
   * **Purpose**: Analyse CDR data to derive actionable insights and support decision-making.
   * **Activities**: Process CDRs to identify network performance metrics, call traffic patterns, and usage trends.
   * **Tools**: Employ analytics platforms and algorithms for data processing and pattern recognition.

**1.7. ASSUMPTIONS, DEPENDENCIES**

**Assumptions:**

1. **Data Availability**: It is assumed that CDRs will be consistently generated and accessible from telecommunications network elements such as switches, routers, and gateways.
2. **Regulatory Compliance**: Assumption that the necessary legal and regulatory frameworks are in place and adhered to regarding data retention, privacy protection, and security measures for handling CDR data.
3. **Technical Infrastructure**: Assumption that the telecommunications infrastructure supporting data collection, processing, and storage of CDRs is adequately maintained and operational.

**Dependencies:**

1. **Network Integration**: The project depends on seamless integration with telecommunications network components to capture and process CDRs effectively.
2. **Data Processing Tools**: Dependency on robust data analytics and processing tools capable of handling large volumes of CDR data and deriving actionable insights.
3. **IT Support**: Dependencies on IT support for maintaining and troubleshooting technical issues related to CDR data collection, storage, and analysis systems.

**1.8. RISKS**

Identifying and managing risks is crucial for the success of the Call Data Record (CDR) project. The following risks should be considered:

* Unauthorized data access due to security gaps, communication failures between client and server, database issues, user errors, and external dependency disruptions.

**2. DESIGN OVERVIEW**

The Call Data Record (CDR) system is intended to record, store, and handle comprehensive call data. It operates by combining several interconnected parts: a processing engine for batch or real-time analysis of CDRs; reliable storage options in databases tailored for large-volume data; data intake from network components such as switches and routers; and an intuitive query interface for effective retrieval. Call kinds (such as audio or video), call statuses (successful, unsuccessful, etc.), timestamps showing call start, end, and duration, and unique call identifiers are some examples of key data fields. Strict access controls, data encryption, and adherence to legal requirements are all included in security measures.

**2.1. DESIGN OBJECTIVES**

**2.1.A** Establish a connection between the client and the server.

**2.1.B** Allow users to sign up for an account and store their information in a database.

**2.1.C** Process user requests which include signing up, logging in, exiting the system, processing CDR files, printing or searching billing information, and logging out.

**2.1.D** Handle customer billing and inter operator settlement billing.

**2.1.E** Terminate the connection upon user request.

**2.1.1. Recommended Architecture**

Based on the context diagram provided, a client-server architecture is recommended for the system. In this setup, there are distinct roles: clients and servers. Clients, such as user devices or interfaces, initiate requests for data or actions. Servers, which are powerful machines, handle tasks like managing databases, processing CDR files, and performing billing operations. This architecture is ideal because it efficiently centralizes complex operations on the server side, where robust processing capabilities are available. Clients, meanwhile, focus on providing user-friendly interfaces for interacting with the system. This clear division of responsibilities ensures efficient data management, secure processing, and seamless user interaction, making it well-suited for the depicted system requirements.

**2.2. ARCHITECTURAL STRATEGIES**

* **Modular Design:** The system can be divided into modules that perform specific tasks, such as user management, authentication, billing, and CDR processing. This would improve maintainability and reusability of the code.
* **Layered Architecture:** The system can be designed with a layered architecture, where each layer provides a specific service to the layer above it. This would improve separation of concerns and make the system easier to understand and modify.
* **Client-Server Architecture:** The system can be designed with a client-server architecture, where the client application interacts with the server application to process user requests and access data. This would distribute processing tasks and make the system more scalable.
* **Database Management System (DBMS):** The system can utilize a DBMS to store user information, billing data, and CDR files. This would ensure data integrity and facilitate efficient data retrieval.
* **Thread Management:** The system can implement thread management techniques to handle multiple client requests concurrently, especially for tasks like CDR file processing and billing which might be resource intensive.

**2.2.1. Creation of New Common Services/Utilities**

* **User Management:** This service could handle user signup, login, and potentially password reset functionalities. It would interact with the database to store and retrieve user information.
* **Authentication:** This service would be responsible for verifying user credentials during login.
* **Database Interaction:** This service could encapsulate all database interactions such as storing user information, CDR files, billing information etc.
* **CDR Processing:** This service could be responsible for processing CDR (Call Detail Record) files, which likely involves parsing the file, extracting relevant data, and potentially storing it in the database.
* **Billing:** This service could handle customer billing and interoperator settlement billing. It might interact with the database to fetch customer information and CDR data.
* These are just a few examples, and the specific common services/utilities that would be created would depend on the specific needs of the system.

**2.2.2. User Interface Paradigms**

It could be a text-based command-line interface (CLI) where you interact through Putty with commands. Alternatively, the server might have a separate web interface you access through a web browser, not Putty. Less likely, the server application could have its own custom graphical user interface (GUI) accessible via Putty. Ultimately, the interface paradigm hinges on the design of the server application for your CDR project.

**2.2.3. Error Detection / Exceptional Handling**

**Client-side:**

* **Connection errors:** The client could detect errors during connection establishment, such as the server being unavailable or unreachable. It could handle this by retrying the connection or notifying the user.
* **Invalid user input:** The client could detect invalid user input, such as an empty username or password during signup. It should provide informative error messages to the user and allow them to correct their input.
* **Network errors:** The client could encounter network errors during communication with the server. It could handle this by retrying the request or informing the user about the issue.

**Server-side:**

* **Database errors:** The server could encounter errors while interacting with the database, such as connection errors or invalid queries. It should handle these errors gracefully to avoid crashing and log them for further investigation.
* **Authentication errors:** The server could detect invalid login credentials or unauthorized access attempts. It should handle these by displaying appropriate error messages and potentially locking accounts after a certain number of failed attempts.
* **Invalid requests:** The server could receive invalid requests from the client, such as requests with missing or malformed data. It should handle these by returning appropriate error messages to the client, indicating what went wrong.
* **Unexpected errors:** The server could encounter unexpected errors during processing, such as software bugs or hardware failures. It should implement proper error handling mechanisms to log these errors, prevent crashes, and potentially initiate recovery procedures.

**2.2.4. Concurrency and Synchronization**

The system can handle multiple client requests simultaneously, likely through multithreading, for improved performance. This allows concurrent processing of functionalities like "Sign Up" and "Login." However, synchronization might be needed in scenarios like concurrent user registrations with the same username or updating billing information. This ensures data consistency by using mechanisms like mutexes or semaphores to control access to shared resources.

**3. SYSTEM ARCHITECTURE**

**3.1. SYSTEM ARCHITECTURE DIAGRAM.**

A diagram of a data structure

Description automatically generated

A screenshot of a computer

Description automatically generated

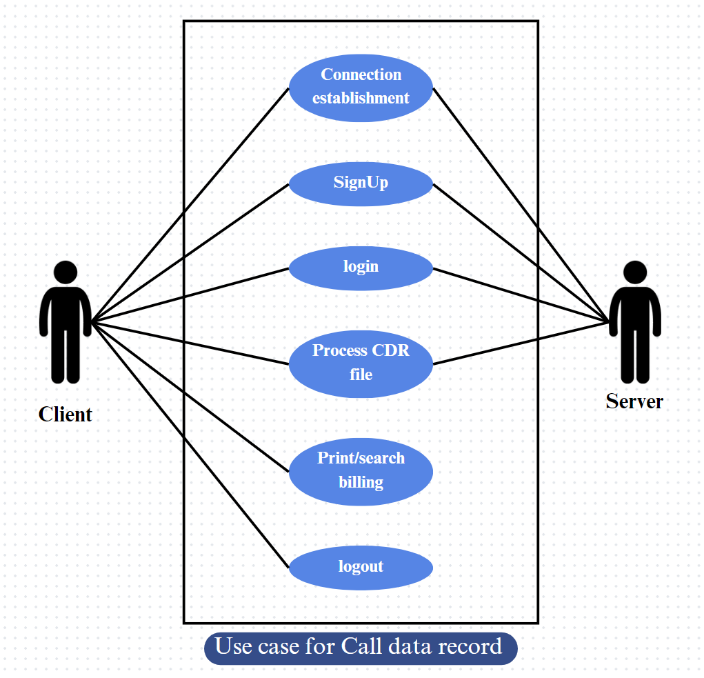
A screenshot of a computer

Description automatically generated

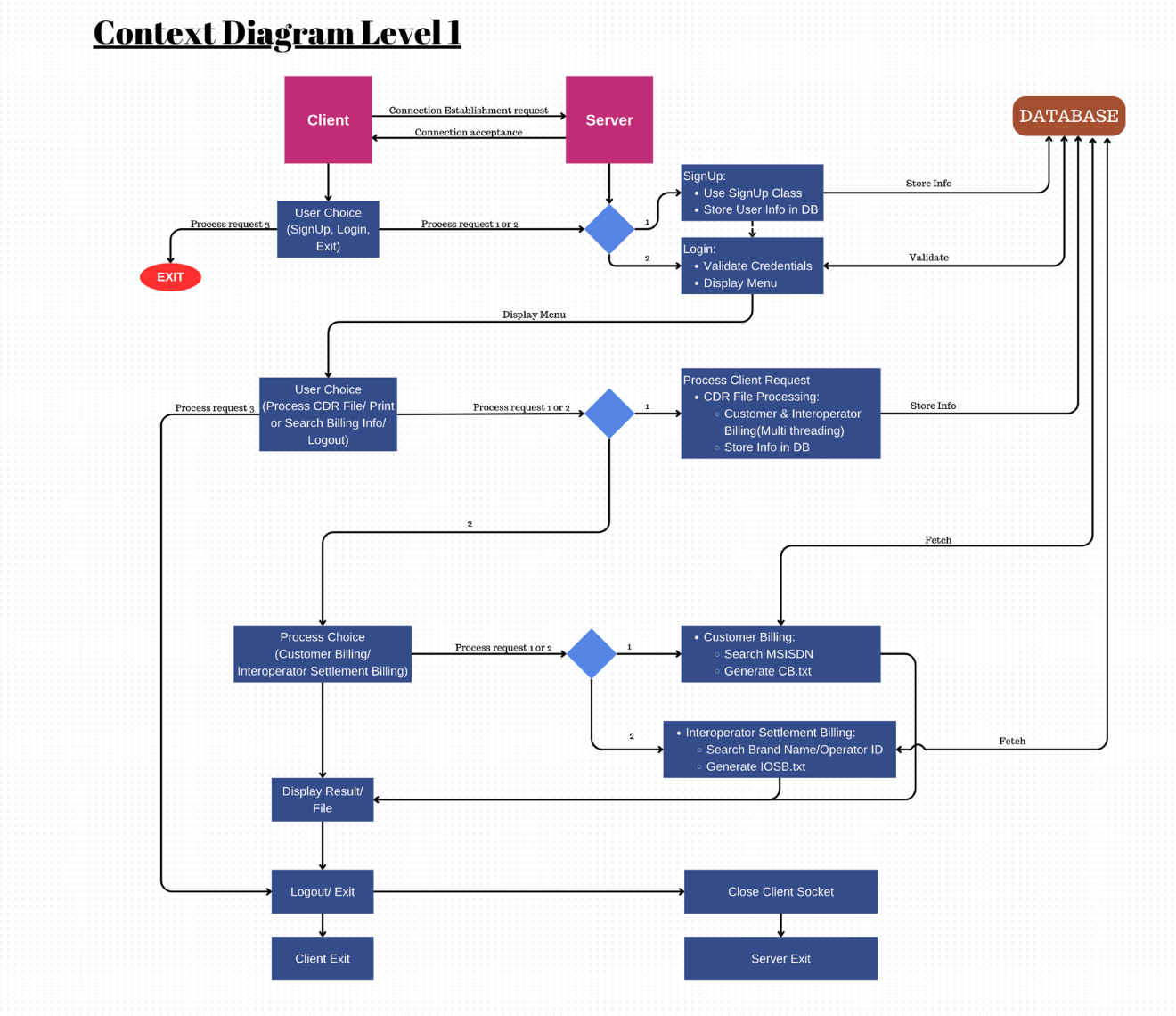
A screenshot of a computer screen

Description automatically generated

**3.2. SYSTEM USE-CASES**



**3.3. SUBSYSTEM ARCHITECTURE**



**3.4. SYSTEM INTERFACES**

**3.4.1. Internal Interfaces**

As an Internal interface we are using Putty Linux Distribution. It is an Operating System made up of software collection based on Linux Kernel, or you can say that distribution contains the Linux Kernel and Supporting libraries and software.

**3.4.2. External Interfaces**

1. Desktop or Linux system

2. Internet

**4. DETAILED SYSTEM DESIGN**

**4.1. KEY ENTITIES**

* **Client:** This represents the user or application that interacts with the system. In this case, it could be a customer or an inter operator.
* **Server:** This is the core of the system that processes requests from the client and generates responses.
* **Database (DB):** This stores information about users, billing data, and other system data.

**4.2. DETAILED-LEVEL DATABASE DESIGN**

The database will likely consist of the following tables:

* **User** table: This table would store information about the users of the system, such as their username, password, and any other relevant user information.
* **CDR** table: This table would store information about the Client Data Reports, such as the date, time, and content of the report.
* **Billing** table: This table would store information about billing, such as customer ID, billing amount, and date.

The tables would have relationships between them. For example, the User table might have a one-to-many relationship with the CDR table, meaning that one user can have many CDRs associated with them. The User table might also have a one-to-many relationship with the Billing table, meaning that one user can have many billing records associated with them.

The specific design of the database would depend on the specific needs of the system. However, the use case diagram provides a good starting point for designing a database that can support the system's functionality.

Here are some additional details that could be included in the database design:

**User\_details.txt:**

* Username
* Password

**CDR table:**

* MSISDN
* Operator Brand Name
* Operator MNC/MMC ID
* Call type
* Duration
* Download
* Upload
* Third party MSISDN
* Third Party Operator MMC/MNC ID

**IOSB.txt**

* Incoming Voice Call Duration
* Outgoing Voice Call Duration
* Incoming SMS
* Outgoing SMS
* MB Downloaded
* MB Uploaded

**CB.txt**

* Service within the operator
  + Incoming Voice Call Durations
  + Outgoing Voice Call Durations
  + Incoming SMS messages
  + Outgoing SMS messages
* Service Outside the mobile operator
  + Incoming Voice Call Durations
  + Outgoing Voice Call Durations
  + Incoming SMS messages
  + Outgoing SMS messages
* Internet Use
  + MB Downloaded
  + MB Uploaded

**4.2.1. Data Mapping Information**

The system interacts with a client and a database. The client sends a connection establishment request to the server. The server can accept the connection or reject it. Once the connection is established, the client can choose to sign up for a new account or login to an existing account.

If the client chooses to sign up, the system will process the signup request and store the user information in the database.

If the client chooses to login, the system will validate the credentials and upon successful validation, display the menu. The client can then choose to process a CDR file, print/search billing information or logout.

The system can process the CDR file, which involves customer and interoperator billing data. It can also generate customer billing and interoperator settlement billing reports. The data is then stored in the database.

**4.2.2. Data Conversion**

* **Client SignUp:** The system converts the user-entered sign-up information into a format suitable for storage in the database. This may involve converting strings to a specific data type for fields like phone number or converting dates into a standard format.
* **Process CDR File:** The system converts the CDR file, which is likely in a specific format (e.g., CDR), into a format suitable for processing and storing the data in the database.
* **Process Client Request (Print/Search Billing):** The system retrieves billing information from the database and converts it into a format suitable for display (on screen) or printing.

**4.3. DISASTER AND FAILURE RECOVERY**

The system should:

* Implement backup and recovery mechanisms to protect against data loss.
* Maintain redundant copies of critical data to ensure fault tolerance.
* Define procedures for restoring data in the event of system failures or disasters.

**4.4. BUSINESS PROCESS WORKFLOW**

* Client initiates a connection and sends a request to the server: The request can be one of three options: Signup, Login, or Exit.
* The server processes the request.
* The client receives the response from the server and performs an action.
* The server processes the client's choice.

**4.5. BUSINESS PROCESS MODELING AND MANAGEMENT**

* **Client Interactions:** Clients initiate connections, which the server accepts. Clients can then perform various actions like Signup, Login, processing/printing a CDR file, searching billing information, or logging out.
* **Business Process Optimization:** Business Process Modelling and Management (BPM&M) can be used to analyse the DFD and identify areas for improvement.
* **Process Improvement and Monitoring:**  By leveraging BPM&M, bottlenecks can be identified and addressed through process flow redesign. Additionally, performance metrics can be implemented to monitor the system's efficiency.

**4.6. BUSINESS LOGIC**

* **Client interacts with server:**  A client of the telecommunication company can sign up, login, or exit the billing system.
* **Process CDR or access billing:** Logged-in clients can choose to process a CDR file (generating billing files) or access existing billing information.
* **Search and display:** For accessing billing information, the client searches by MSISDN or operator details. The system retrieves and displays the results, allowing the client to view them or log out.

**4.7. VARIABLES**

**Client-side:**

* Username
* Password
* User choice (e.g., Signups, Login, Exit, Process CDR File/Print, Search Billing Info, Logout)
* MSISDN (Mobile Station Integrated Services Digital Network Number)
* Brand Name
* Operator ID.

**Server-side:**

* Connection status (established/not established)
* Login credentials (valid/invalid)
* Billing information.

**Database:**

* User information (e.g., username, password)
* Billing information.

**4.8. ACTIVITY / CLASS DIAGRAMS**

A diagram of a company

Description automatically generated

**4.9. DATA MIGRATION**

Data migration typically involves transferring data from one system to another. The provided context diagram illustrates a high-level view of a client-server system interaction.

* The client establishes a connection with the server.
* The server validates the user credentials upon login.
* The client can choose various functionalities like signup, login, displaying menu, processing CDR files (Customer Data Records), customer billing or inter operator settlement billing and logout.
* The server processes these requests and interacts with the database.

**4.9.1. Architectural RepresentatioN**

* Users connect to the box and choose what they want to do (Sign up, Login, Exit).
* The box handles those requests (validates user info, processes billing files, etc.).
* The box stores user and billing information in a separate filing cabinet (database).
* Once done, the user disconnects from the box.

**4.9.2. Architectural Goals and Constraints**

**Goals:**

* **Easy to Change:** System parts should be independent, making updates or fixes easier.
* **Handles Growth:**  The system should be able to handle more users and data over time.
* **Keeps Data Safe:**  User information and system access should be protected.

**Constraints:**

* **Fast & Smooth:**  The system should respond quickly and operate efficiently.
* **Always Available:**  The system should be reliable and accessible to users.
* **Easy to Maintain:**  The system should be clear and easy to update when needed.

**4.9.3. Logical ViEW**

**The logical view describes:**

* System decomposition into subsystems and modules.
* Packages and classes representing key functional areas and components.
* Relationships, interfaces, and interactions between system elements.

**4.9.4. Architecturally Significant Design Packages**

* **Client:** This package is responsible for interacting with the server. It sends requests to the server and receives responses. In the context diagram, it initiates the connection establishment request and user choices.
* **Server:** This package is responsible for processing requests from the client. It can perform various operations such as storing and retrieving data from the database, validating user credentials, processing CDR files, and generating billing reports.
* **Database:** This package is responsible for storing and retrieving data. The context diagram shows the database being used to store user information and CDR file information.

**4.9.5. Data model**

The data model translates the context diagram entities (Client, Sign Up, Login, CDR File, Customer Billing, Interoperates Settlement Billing) into relational database tables with attributes like usernames, passwords, billing details, and file content. Foreign keys ensure data consistency by linking related tables (e.g., Sign Up and Login referencing Client usernames).

**4.9.6. Deployment View**

* + 1. **Physical Components:** It shows the physical building blocks like servers, databases, and client machines that run the software.
    2. **Connections:** It depicts how these components are connected. This includes connections between clients and servers for exchanging information and server-database links for data storage and retrieval.
    3. **Deployment Flexibility:** The specific layout (single server or distributed architecture) depends on the system's needs.
    4. **Limited Detail from Context Diagram:** While a deployment view can include hardware/software types, network topology, and security measures, the context diagram itself doesn't provide those specifics. It offers a general understanding of the deployment structure.

**5. ENVIRONMENT DESCRIPTION**

The environment we are using is Linux operating system. The environment of a Linux operating system can be described in several aspects:

**Kernel**: At the core of every Linux system is the Linux kernel. It manages hardware resources, provides essential services for other parts of the operating system, and acts as an intermediary between software and hardware.

Shell: Linux systems typically use a shell interface for interacting with the operating system.

**File System**:Linux employs a hierarchical file system structure. The root directory ("/") is the top-level directory, and everything in the system is organized beneath it. Common directories include "/bin" (executables), "/etc." (configuration files), "/home" (user home directories), "/var" (variable data), and many others.

Graphical User Interface (GUI): While Linux is often associated with command-line interfaces, many distributions offer GUI environments as well.

**Package Management**: Linux distributions typically use package managers to install, update, and remove software.

**Networking**: Linux provides robust networking capabilities, supporting various networking protocols and services.

**Security**: Linux is known for its strong security features. This includes file permissions, user authentication mechanisms like passwords and SSH keys, firewalls.

**Multitasking and Multiuser**: Linux is a multitasking and multiuser operating system, meaning it can run multiple processes simultaneously and support multiple users accessing the system concurrently. Each user has their own user account and home directory, and permissions are enforced to protect system resources.

**Command-Line Tools and Utilities**: Linux offers a vast array of command-line tools and utilities for various tasks such as text processing, system administration, programming, networking, and more.

**5.1. LANGUAGE SUPPORT**

C language is used in this project. It was created in the 1970s by Dennis Ritchie and remains very widely used.

**5.2. USER DESKTOP REQUIREMENTS**

* Windows: 7 or above
* Processor: Minimum 1GHz and more
* Hard Drive: 32GB and more
* Memory (RAM): Minimum 1GB and more

**5.3. SERVER-SIDE REQUIREMENTS**

* **Database**

The server will need a database to store user information, CDR (Call Detail Record) files, and billing information. The size of the database will depend on the number of users and the amount of data that is being stored.

* **Application Server**

The server will need an application server to run the software that processes user requests. The application server will need to be able to handle multiple concurrent users.

* **Network**

The server will need a network connection to communicate with clients. The bandwidth of the network connection will depend on the amount of data that is being transferred.

**5.3.1. Deployment Considerations**

* **Scale up or out:** Handle many users by adding power or distributing workload.
* **Fortress approach:** Secure data with encryption, authentication, and authorization.
* **Fast response:** Optimize code, database, and cache frequently used data.
* **Always-on mentality:** Use redundancy and disaster recovery for high availability.
* **Easy updates:** Modular code and documented procedures for smooth maintenance.
* **Mind the budget:** Choose a cost-effective solution for hardware, software, and upkeep.
  + 1. **Integration Requirements**
* **Client-Server:** The client needs to be able to establish a connection with the server. The server needs to be able to accept connections from clients.
* **Client-Database:** The client needs to be able to access the database to perform operations such as signup, login, and customer billing.
* **Server-Database:** The server needs to be able to access the database to store user information, process CDR files, and perform billing operations.

**5.3.3. Operating System**

This software could run on various platforms like Windows, Linux, macOS, or even specialized server operating systems.

**5.3.4. Desktop**

We are using Linux operating system and we are working through putty.

**6. REFERENCES**

|  |
| --- |
| 1. <https://docstore.mik.ua/univercd/cc/td/doc/product/wanbu/das/das_1_4/das14/das14apd.htm> |
| 1. <https://www.gl.com/Presentations/Call-Data-Records-Presentation.pdf> |
| 1. <https://en.wikipedia.org/wiki/Call_detail_record> |
|  |
| 1. <https://gist.github.com/kaisesha/bd10fd299a3bed2b12ff031c937cdd4c> |
| 1. <https://github.com/kaisesha/cdrgraph/blob/master/CDR_Gist3.csv> |
| 1. <https://anjuchamantha.github.io/cellyzer---CDR-data-analyzer/> |
| 1. <https://github.com/mayconbordin/cdr-gen> |
| 1. <https://github.com/deshpandetanmay/cdr-data-generator> |
| 1. <https://www.sciencedirect.com/topics/computer-science/call-data-record> |
| 1. <https://www.itu.int/en/ITU-D/Emergency-Telecommunications/Documents/2017/Reports/LB/D012A0000C93301PDFE.pdf> |
| 1. <https://ijarcce.com/upload/2016/december-16/IJARCCE%2064.pdf> |
| 1. <https://www.etsi.org/deliver/etsi_ts/122100_122199/122115/03.02.00_60/ts_122115v030200p.pdf> |
| 1. <https://lawwatch.in/how-to-obtain-call-data-records-cdr/> |
| 1. <https://www.etsi.org/deliver/etsi_ts/132200_132299/132298/17.03.00_60/ts_132298v170300p.pdf> |

**7. APPENDIX**

* C functions
* Pointers
* File handling
* Socket Programming
* Structures