PRD:Customer Data Platform

Introduction : We aim to build a Customer Data Platform (CDP) that acts as a single source of truth for all customer-related data by integrating multiple internal and external systems. Currently, customer data is scattered across CRM systems, Website DB, Ad Platforms, Call Center, CPaaS, Engagement App, Product Analytcs, ERP tools, Ticketing software etc.

In Phase 1, the focus will be on integrating CRM data and call recording systems. This will establish the foundation of the centralized data layer. In future phases, we will enhance the CDP by integrating behavioral, website, and message/event data to enable advanced analytics and personalization at scale.

Problem Statement: Currently, we face several operational and data challenges that limit marketing efficiency, sales productivity, and student experience:

2.1 Lack of Unified View of Student

- Data is fragmented across CRM, call recordings, WebEngage, and internal systems.
- There is no single profile that captures a student's complete journey (calls, enrollments, visits, eligibility, objections, etc.).

2.2 Basic Segmentation Is Not Possible

The marketing team is currently unable to build basic segments, leading to wasted spend and missed opportunities. Some examples of required segmentation that are not possible today:

- Segment of students who gave ANTHE this year, qualified, but haven't been contacted yet.
- Segment of students eligible to pitch for RCC courses.
- Segment of students who have ever spoken to a branch person or counselor about AD, DLP, or RCC.
- Build a segment → students who had a counselor conversation in the last 90 days.
- Build an segment → students whose "reason for not joining" was fees.

2.3 Data Issues & Inconsistencies in CRM

CRM data pushed into WebEngage contains inconsistencies and formatting issues. Missing fields, wrong value types, or undefined mappings make segmentation and automation difficult.

2.4 No Visibility on Course Eligibility

- Currently, we cannot reliably see which student is eligible for which course (e.g., RCC eligibility, ANTHE qualifiers, academic year status).
- This limits targeted communication and counselor productivity.

Goal of Phase 1:

The main objective of Phase 1 of the CDP project is to integrate CRM data and call recordings to establish a foundational, clean, and structured data layer. This will include both incremental and historical/back data of the Students, which can then be exported to MoEngage or WebEngage depending on the business requirements. After this marketing and business team would be able to do basic segmentation around all the Student Details which we are going to list. All the Segementation Usecases are listed here

Data Pipeline Requirements for CDP (Snapshot → Silver → Gold):

This section defines how Superleap data must be ingested, cleaned, modeled, and stored across the data layers (Snapshot, Silver, and Gold) to support the CDP architecture.

1. Create Clean Snapshot Tables (Bronze Layer)

We will maintain *clean, standardized snapshots* of all Superleap source tables before using them to build any CDP tables.

Objective

- Ensure consistent schema, correct data types, and removal of non-standard null formats.
- Act as a stable middle layer between raw Superleap data and CDP transformations.

Technical Requirements

- Table naming pattern: superleap_prod.<table>_clean_snapshot

  Example:
  Raw → superleap_prod.lead
  Snapshot → superleap_prod.lead_clean_snapshot

- Data type alignment:
  - Each field must be cast to the correct business-use data type (e.g., dates to date, phone to string, IDs to string, amounts to decimal, flags to boolean).
- Null handling:
  - All empty strings ("") or literal "null" strings must be converted to SQL NULL.

Expected Output:

A set of “clean snapshot” tables that serve as the single, reliable input layer for all Silver and Gold CDP transformations.

---

2. Build Silver Layer for Pre-admission Customer Data

The Silver Layer will transform cleaned snapshots into modeled, dimensional data stored in existing Athena datarepository tables.

Objective

- Normalize, standardize, and enrich Superleap pre-admission data.
- Maintain granular, dimension/fact aligned schemas.

Technical Requirements

1. Silver-layer tables must integrate with existing databases:
   - datarepository_dimension_tables
   - datarepository_fact_tables
2. Granularity:
   - Dimension tables store **entity-level data** (lead, branch, program, user, etc.)
   - Fact tables store **event/transaction-level data** (lead actions, status changes, calls, etc.)
3. Example::
   - A branch name from Superleap should first be added to the existing Crm_dim_branch and then used to create the final user and events data in the gold layer
4. Only **clean snapshot tables** should be used for transformation.

---

3. Create a Dedicated CDP Gold Layer (Final Output)

A new database will be created in Athena to store the unified CDP system-of-record output.

Objective

- Maintain Users and Events tables.
- These tables will be integrated with tools like MoEngage or WebEngage (as per business requirements) to support segmentation and targeting use cases.

CDP Data Model: Silver and Gold Layer Table Inventory :

Silver Layer Tables :-

Dimension Tables

- crm_dim_lead
- crm_dim_user
- crm_dim_city
- crm_dim_state
- crm_dim_school
- crm_dim_branch
- crm_dim_stream

Fact Tables

- crm_fact_scholarship_opportunity
- crm_fact_exam_attendance
- crm_fact_exam_result
- crm_fact_product_opportunity
- crm_fact_course
- crm_fact_call_disposition

Gold Layer (CDP Output) :-

User Attributes

- Lead details
- Current ANTHE details
- Previous ANTHE details
- Current IACST Attempt 1 details
- Current IACST Attempt 2 details
- Previous IACST Attempt 1 details
- Previous IACST Attempt 2 details
- Current IAT details
- Previous IAT details
- Latest Product Opportunity details
- Previous Product Opportunity details

Event Attributes

- Call disposition details

Catalog

- Branch details

The Detailed sheet of all the fields are attached here

Logic for all the Calculated Fields:

All the Calculated field are being marked as red in the Detailed sheet of all the fields I have attached

1. Logic for Deriving Messaging Mobile Numbers :

The Superleap database have six different mobile numbers stored for a lead:

1. Primary Mobile Number
    - The number from which the lead was originally captured.
    - Can belong to the student, father, or mother.
2. Student Mobile Number
    - Explicitly provided as the student's own number.
3. Father Mobile Number
    - Explicitly provided as the father's number.
4. Mother Mobile Number
    - Explicitly provided as the mother's number.
5. Alternate Mobile Number
    - Sourced from bulk uploads.
6. Principal Mobile Number
    - Sourced from bulk uploads of school data.

Objective is to create three derived fields that clearly identify who is being messaged, so marketing communication is sent to the correct recipient:

1. student_msg_no
2. father_msg_no
3. Mother_msg_no

A. Logic for `father_msg_no` and `mother_msg_no`

Population Conditions:Because most important for is the student's mobile number if that is available to us then we will care about father and mother mobile number

- `Father_msg_no` is populated only if:

    - Father Mobile Number exists, and
    - It is not equal to Student Mobile Number, and
    - It is not equal to Primary Mobile Number, and
    - Student Mobile Number and Primary Mobile Number are both present.
- `mother_msg_no` follows the exact same logic, using Mother Mobile Number.

Otherwise

- If any of the above conditions fail, the respective field is set to NULL.

Logic for `student_msg_no`:

`student_msg_no` represents the **primary number to reach the student** and is derived using the following priority order:

1. **If Student Mobile Number is present**
   → `student_msg_no = student_mobile_number`
2. **Else if Student Mobile Number is not present and Primary Mobile Number exists**
   → `student_msg_no = primary_mobile_number`
3. **Else if neither Student nor Primary Mobile Number is present**
   → `student_msg_no = father_mobile_number or mother_mobile_number whatever is available`

---

2. Logic for Father and Mother Name in User Attributes

Step 1: Source of Truth

- Source table: crm_dim_lead (Silver Layer)
- Source fields:
    - father_name
    - mother_name

These fields may contain full names (single or multiple words).

Step 2: Name Standardization

Each parent's name is split into first name and last name before storing in the Gold Layer.
Target user attributes:

- father_first_name

- father_last_name
- mother_first_name
- mother_last_name

Step 3: Splitting Logic

First Name

- The first word of the full name string.

Last Name

- All remaining words after the first word, concatenated with a single space.

Example

- father_name = "Rajesh Kumar Sharma"
  - father_first_name = "Rajesh"
  - father_last_name = "Kumar Sharma"

Step 4: Edge Case Handling

- If the full name contains only one word:
  - *_first_name = full name
  - *_last_name = NULL

- If the full name is NULL or empty:
  - Both *_first_name and *_last_name are set to NULL

- Leading/trailing spaces are trimmed before processing.

Step 5: No Fallbacks

- No fallback from other tables or fields.
- Names are populated only from crm_dim_lead.

---

3. Logic for Current and Previous Scholarship Opportunities

Step 1: Identify Current and Previous Scholarship Opportunities

For each student, scholarship opportunities are first grouped by academic year and then ordered by created_date (descending).

Current Scholarship Opportunity

→ Scholarship opportunity with the most recent created_date within the current academic year.

Previous Scholarship Opportunity

→ Scholarship opportunity with the most recent created_date within the immediately previous academic year.

Note:

- Current = most recent in current academic year
- Previous = most recent in previous academic year

This logic is applied independently per exam type.

Step 2: Split by Exam Type

Scholarship opportunities are categorized using exam_type from crm_fact_scholarship_opportunity (Silver Layer).

Supported exam types:

- ANTHE
- IAT
- IACST

Step 3: Define Current and Previous Fields per Exam

ANTHE

- current_anthe_
  → Latest ANTHE scholarship opportunity
  (most recent created_date in the current academic year)

- previous_anthe_
  → Previous ANTHE scholarship opportunity
  (most recent created_date in the previous academic year)

IAT

- current_iat_
  → Latest IAT scholarship opportunity
  (most recent created_date in the current academic year)

- previous_iat_
  → Previous IAT scholarship opportunity
  (most recent created_date in the previous academic year)

IACST (Multiple Attempts Supported)

A student can attempt the IACST exam multiple times within the same academic year (maximum two attempts per academic year).

Attempt Identification Logic

- Filter IACST scholarship opportunities to the relevant academic year.
- Order records by created_date descending.
- Assign a rank using this order:
    - rank = 1 → Most recent IACST attempt
    - rank = 2 → Second most recent IACST attempt

This ranking is recalculated independently for each academic year.

Current IACST Fields (Current Academic Year)

- current_iacst_1_
  → IACST record with rank = 1
  (most recent IACST attempt in the current academic year)

- current_iacst_2_
  → IACST record with rank = 2
  (second most recent IACST attempt in the current academic year)

If only one IACST attempt exists in the current academic year:

- current_iacst_1_ is populated
- current_iacst_2_ is NULL

Previous IACST Fields (Previous Academic Year)

- previous_iacst_1_
  → IACST record with rank = 1
  (most recent IACST attempt in the previous academic year)

- previous_iacst_2_
  → IACST record with rank = 2
  (second most recent IACST attempt in the previous academic year)

If only one IACST attempt exists in the previous academic year:

- previous_iacst_1_ is populated
- previous_iacst_2_ is NULL

Step 4: Missing Data Handling

- If a student has only one scholarship opportunity for an exam:
    - current_* is populated
    - previous_* is set to NULL
- If an exam type does not exist for a student:
    - All related fields are NULL.

---

4. Logic for Previous Exam Given Flags

Target Fields:

previous_anthe_given
previous_anthe_given
previous_iacst_1_is_givenx
previous_iacst_2_is_given

Source of Truth:

- Table: crm_fact_scholarship_opportunity
- Key fields used:
    - student_id / lead_id
    - exam_type
    - created_date

Field-wise Logic:

previous_anthe_given

- Condition:
    - exam_type = 'ANTHE'
    - year in created_date belongs to the previous academic year
- Result:
    - If any such product opportunity exists → Yes
    - Else → No

previous_iat_is_given

- Condition:
    - exam_type = 'IAT'
    - year in created_date belongs to the previous academic year
- Result:
    - If any such product opportunity exists → Yes
    - Else → No

previous_iacst_1_is_given

- Condition:
    - exam_type = 'IACST'
    - year in created_date belongs to the previous academic year
    - Attempt ranking (by created_date descending) = rank 1
- Result:

    - If a rank-1 IACST product opportunity exists → Yes
    - Else → No

previous_iacst_2_is_given

- Condition:
    - exam_type = 'IACST'
    - term belongs to the previous academic year
    - Attempt ranking (by created_date descending) = rank 2
- Result:
    - If a rank-2 IACST product opportunity exists → Yes
    - Else → No

---

5. Logic for Latest and Previous Product Opportunities

- **Latest Product Opportunity**
  The product opportunity with the most recent `created_date` for a given student.

- **Previous Product Opportunity**
  The product opportunity created immediately before the latest one for the same student (i.e., the second most recent `created_date`).

---

6. Logic for User Attributes Based on Main Product Opportunity

Attributes to derive in the CDP Gold Layer:

- **term**
- **class_studying_in**
- **course_code**
- **career_interested_in**
- **stream**

Priority Order for Attribute Population

- Priority 1 → If student has Product Opportunity, always use Product Opportunity logic
- Priority 2 → If student does not have Product Opportunity, use Scholarship → Lead fallback
- The moment a Product Opportunity is created, the system will stop fallback usage and permanently follow Priority 1 logic.

CASE 1 — Student has at least one Product Opportunity (Priority 1)

Use Latest Valid Product Opportunity.

- **Stream**
  - `stream = latest_product_opportunity.stream`
- **Term**
  - `term = latest_product_opportunity.term`
- **Class Studying In**

  Uses April rollover logic:

  - If current date < 1 April of Term Start Year
    → `class_studying_in = latest_product_opportunity.class_studying_in`
  - If current date ≥ 1 April
    → `class_studying_in = latest_product_opportunity.class_studying_in + 1`

    Example: Present Year = 2025

  - If today is before 1 April 2024:
    class_studying_in = original value

  - If today is on or after 1 April 2024:
    class_studying_in = original value + 1

- **Course Code** - Take course_code directly from the latest product opportunity.

  - `course_code = latest_product_opportunity.course_code`

- **Career Interested In** - Take career_interested_in directly from the latest product opportunity.

  - `career_interested_in = latest_product_opportunity.career_interested_in`

- **Missing Fields Handling**

  - If any Product Opportunity attribute is null → keep NULL

  - Do not fallback to scholarship or lead once product opp exists

CASE-2 — Student DOES NOT Have any Product Opportunity (Priority 2)

Stream:

Priority order:

1. If the student has applied for a scholarship opportunity:
   Use the *stream* mentioned in the scholarship opportunity.

2. If the student has not applied for any scholarship opportunity:
   Use the *stream* from the lead table.

3. If the stream is not present in the lead table as well:
   Leave the stream field empty/null.

Class Studying In :

Source → Scholarship Opportunity
If null → `NULL`

If available → apply April rollover

- If current date < 1 April of Scholarship Term Start Year
  → `class_studying_in = scholarship.class_studying_in`

- If current date ≥ 1 April
  → `class_studying_in = scholarship.class_studying_in + 1`

Career Interested In :

Default system pitch course

- `career_interested_in = 'RCC'`

Course Code:

- `course_code = NULL`

Term:

- `term = NULL`

Academic Year Logic (Applies Wherever Class Increment Needed)

- Academic rollover date = 1 April
- Increment only once per academic cycle
- Never decrement
- No multi-year retrospective recalculation

Behavior on Transition

If a user moves from no Product Opportunity → has Product Opportunity

- System immediately switches to CASE 1 logic

- Previously derived scholarship / lead values are ignored going forward

Final Output :

- Values stored in Gold Layer
- Empty values must be `SQL NULL`

---

7. Logic for the Attendance data

Target Fields:

- current_anthe_submitted_date, current_anthe_no_of_mock_taken, current_anthe_is_sample_taken, current_anthe_sample_submitted_date, current_anthe_mock_submitted_date, previous_anthe_exam_submitted_date

- current_iacst_1_submitted_date, current_iacst_1_no_of_mock_taken, current_iacst_1_is_sample_taken, current_iacst_1_sample_submitted_date, current_iacst_1_mock_submitted_date, previous_iacst_1_submitted_date

- current_iacst_2_submitted_date, current_iacst_2_no_of_mock_taken, current_iacst_2_is_sample_taken, current_iacst_2_sample_submitted_date, current_iacst_2_mock_submitted_date, previous_iacst_2_submitted_date

- current_iat_submitted_date, current_iat_no_of_mock_taken, current_iat_is_sample_taken, current_iat_sample_submitted_date, current_iat_mock_submitted_date, previous_iat_submitted_date

  Source Table:
- **Attendance Table:** `crm_fact_exam_attendance` (Superleap – Silver Layer)
- **Opportunity Table:** `crm_fact_scholarship_opportunity`

Key Field

- Test_type__c -
  Possible values: mock , sample , scholarship
- exam_date__c
  Description: Date on which the exam was scheduled or conducted
- action__c
  Possible values: `Submitted`, `Auto Submitted`, `Not Attended`

Pre-Condition: Correct Student and Opportunity Mapping

Before deriving any exam-related attributes, ensure exam records are mapped to the correct student and scholarship opportunity.

Step 1: Join Logic

- Join crm_fact_exam_attendance with crm_fact_scholarship_opportunity on opportunity_associated and opportunity_id respectively

Step 2: Identify the Student

- Use associated_lead from crm_fact_scholarship_opportunity to identify the correct student (lead).

Validation Rule

- An exam record is considered **valid** only if:
    - It maps to the same **scholarship opportunity**, and
    - It belongs to the same **associated_lead (student)**.

This ensures exam attendance data is not mixed across:

- different students, or
- different scholarship opportunities.

Lets take current_anthe_fields for now and the same logic would be applied to other exams

CASE 1: Mock Tests

Fields Covered:

- current_anthe_no_of_mock_taken
- current_anthe_mock_submitted_date

Field: `current_anthe_no_of_mock_taken`
Definition : Represents the number of mock tests successfully attempted by the student.

Logic : Count records where -

- test_type__c = 'mock'
- action IN ('Submitted', 'Auto Submitted')

Output Rules

- Valid range: 0 to 4 (maximum 4 mocks exist).
- If no qualifying records exist:
    - current_anthe_no_of_mock_taken = 0

Field: current_anthe_mock_submitted_date

Definition: Captures the most recent submission date among all ANTHE mock tests attempted.

Logic:
If any record exists where:

- test_type__c = 'mock'
- action__c IN ('Submitted', 'Auto Submitted')

→ Populate with the **maximum action_time__c** across all mock submissions
Else
→ Leave the field NULL

CASE 2: Sample Test

Field : current_anthe_is_sample_taken

Definition : Indicates whether the student has attempted the Sample Test.

Logic : If **any** record exists where -

  - test_type__c = 'sample'
  - action IN ('Submitted', 'Auto Submitted')

→ current_anthe_is_sample_taken = Yes

- Else
  → current_anthe_is_sample_taken = No

Field: current_anthe_sample_submitted_date

Definition:
Captures the submission date of the ANTHE Sample Test.

Logic:
If any record exists where:

- test_type__c = 'sample'
- action__c IN ('Submitted', 'Auto Submitted')

→ Populate with the **latest action_time__c** for sample test
Else
→ Leave the field NULL

CASE 3: Scholarship Exam Submission

Field : current_anthe_exam_submitted_date

Definition : Indicates whether the main scholarship exam was submitted and captures the submission date

Logic : If **any** record exists where -

- test_type__c = 'scholarship'
- action IN ('Submitted', 'Auto Submitted')

→ Populate action_date_time as current_anthe_is_exam_submitted_date

- Else
  → Leave the field **NULL**

Field : previous_anthe_exam_submitted_date

Make sure the opportunity_id you would be using here is from the previous scholarship opportunity of the student

Definition : Indicates whether the previous scholarship exam was submitted and captures the submission date

Logic : If **any** record exists where -

- test_type__c = 'scholarship'
- action IN ('Submitted', 'Auto Submitted')

→ Populate action_date_time as previous_anthe_is_exam_submitted_date

- Else
  → Leave the field **NULL**

8. Logic for all the additional fields

Target Fields:

Last 45 Days

- last_45_days_ad_remark
- last_45_days_dlp_remark
- last_45_days_rcc_remark
- last_45_days_branch_visit_no
- last_45_days_home_visit_no
- last_45_days_connected_call_no
- no_of_times_lead_lost_45_days
- lead_lost_region_45_days
- no_of_times_lead_lost_region_45_days

This Academic Year

- this_academic_year_ad_remark
- this_academic_year_dlp_remark
- this_academic_year_rcc_remark
- this_academic_year_branch_visit_no
- this_academic_year_home_visit_no
- this_academic_year_connected_call_no
- no_of_times_lead_lost_this_academic_year
- lead_lost_region_this_academic_year
- no_of_times_lead_lost_region_this_academic_year

Lifetime

- lifetime_ad_remark
- lifetime_dlp_remark
- lifetime_rcc_remark
- lifetime_branch_visit_no
- lifetime_home_visit_no
- lifetime_connected_call_no
- no_of_times_lead_lost_lifetime
- lead_lost_region_lifetime

no_of_times_lead_lost_region_lifetime

Other User Attributes and Events -

These are the user attributes that has to be directly picked from the following tables

A. Lead Table - First Name, Last Name, Student mobile No, LMS Created By, Gender, Birth_date, superleap_origin, created by, Created date, stream, PSID, Category, whatsapp opt in, Class Topper, Lead Source, UTM Campaign, UTM ID, UTM Source, UTM Medium, campaign id, campaign name, Father's Name - split into first and last, Mother's Name, Stage, School Exam board, otherSchoolName, email, Principle mobile No , Superleap_lead_id

B. City Table - City, City code, Pincode , Latitude, Longitude

C. School Table - School Name, School City, School Code, School address

D. Branch Table - branch_name, branch_code, Branch zone, Branch city

E. School Table - school id, School Name, School City, School Code, School address

F. Scholarship Opportunity Table - S_1_Stage, S_1_Exam type, S_1_Exam Mode, S_1_Exam Center, S_1_date of exam, S_1_original examination date, S_1_stream, S_1_current term, S_1_class studying in, S_1_Roll number, S_1_Created at, S_1_UTM Id, S_1_UTM Medium, S_1_UTM Campaign, S_1_Campaign id, S_1_UTM Source, S_1_Roll number generation date, S_1_Payment Date, S_1_Payment Mode, S_1_Payment Type, S_1_Transaction Amount, S_1_Transaction ID, S_1_Transaction Status, S_1_Payment Link, Mock taken or not, Mock created date, Mock ID, Payment Date Mock Test, Payment Mode Mock Test, Transaction Amount Mock Test, Payment Mock url

G. Result Table - Percentage, Display Remark, Eligible Scholarship, national rank, state rank, City Rank

H. Product Opportunity - Stage Name, courseCode, opportunity id, course name, Loss Reason, Class Studying In, Next Follow-up date, Payment Date, Payment Mode, Payment Url, Payment Type, Payment Vendor, Source, Stage Sub-disposition, Transaction Amount, Transaction ID, Transaction Status, Application ID, PDF Path, Branch

E. Career Table - Career

You can see the detailed sheet along with the api name of each attribute here

Requirements from Data Lake point of view

Anything which is based on academic year or current term should be kept as a single configurable parameter

1. Create clean snapshots of superleap as a middle layer before using to create CDP tables - data types should be as per business use case. All null values should be SQL nulls instead of empty strings or string nulls.
    a. Superleap_prod.lead -> superleap_prod.lead_clean_snapshot
2. There will be a silver layer for preadmission customer data which will be added to the existing datarepository_dimension_tables and datarepository_fact_tables in Athena. Granularity based table schema will be maintained. For example, a branch name from Superleap should first be added to the existing core_dim_branch and then used to create the final user and events data in the gold layer. Clean superleap tables should be used.
    a. crm_dim_lead
        i. first_name - eg Laksh
        ii. last_name - eg Arora
        iii. superleap_lead_id
        iv. lms_created_by
        v. superleap_origin
        vi. ps_created_by
        vii. campaign_id
        viii. term
        ix. created_by
        x. city_id
        xi.

3. Create a separate database in Athena for CDP gold layer with tables
    a. User
        i. first_name - crm_dim_lead.first_name
        ii. user_id - superleap_lead_id
        iii. lead_origin - one of Website, Branch, Call Center, External
        iv. Is_bulk_upload -
        v. campaign_name
        vi. created_by - crm_dim_user.name
        vii. city - crm_dim_city.name
        viii. class_studying - if current date < 1 apr, add year difference between current year and lead created date
        ix. class_opted
        x. last_name - crm_dim_lead.last_name
        xi. superleap_lead_id - crm_dim_lead.superleap_lead_id
        xii. lms_created_by - crm_dim_lead.lms_created_by
        xiii. superleap_origin - crm_dim_lead.superleap_origin
        xiv. ps_created_by - crm_dim_lead.ps_created_by
        xv. campaign_id - crm_dim_lead.campaign_id
        xvi. term - crm_dim_lead.term
        xvii. created_by - crm_dim_lead.created_by
        xviii. city_id
        xix. class_studying
        xx. created_date
    b. Event