This document details the Customer Privacy Management system that we will be building as a part of our ongoing project.

**What will you learn**

* System architecture
* Python programming as required in commercial applications
* Securing python code
* Making Python code that performs within SLAs (Service Level Agreements of code - an agreement between customer and solution developer guaranteeing a minimum level of performance)
* Writing code that scales with increasing workload
* Integrating with database
* Building secure APIs in Python

**Platform to be used for build**

**AWS**

Will be arranged and managed by us.

Will be built on the same lines as the Prospareto production cloud setup.

Students will be given access.

We will use:

* RDS for MySQL
* EC2 for Linux compute nodes
* Gunicorn for web server

**Architecture to be created**

MySQL backend

Database access control through MySQL/Data access controlled from Python

Main Python classes serving API requests

Common modules

API Security layer designed in Python

Endpoint layer

Gunicorn web server for handling requests

**Processes**

Requirement gathering

High level design

Low level design

Development

Unit testing

Integration testing

Deployment

Early Life Support (ELS)

Close

**Requirements**

Access through web services.

Customer logs in.

**TASK:**

New customer id is created. It shall be unique.

Customer's login defaults to certain topics subscriptions.

**INPUT FORMAT:**

inp={

"f\_nm":"Iron",

"l\_nm":"Man",

"dob":"",

"g":"m",

"eml":"ironman@avengers.com",

"img":"",

"ph":"1765490324",

"al1":"",

"al2":"",

"al3":"",

"tg":"67682983789"

}

**OUTPUT FORMAT:**

If there are no errors.

out={

"data":{

"idc":7649724562736

},

"error":[]

}

If there are errors.

out={

"data":{},

"error":[200, “Invalid address”]

}

**SUBTASKS:**

Accept incoming data

Check if customer has accessed before using their email address

If yes, then return their customer id

If not, then

generate a new, unique customer id

store customer information the database

store email information in the database

store phone information in the database

store address information in the database

Return newly generated customer id

Convert code to user modular components

**DESIGN/PROCESS FLOW:**

User logs in to the front end - sends a json to the server

Login function is executed on the server

Return is made after processing

FUNCTIONS NEEDED:

generate a new, unique customer id

Do login checks/processing

FOR LATER:

Integrate logging and auditing

Integrate generic error handling

Customers can opt for certain subscriptions for different mediums.

Allow customers to change their subscription details.

Customer wants to fetch their subscribed items list

Customer can update mobile number, email address, home address, image

Customer wants to change subscriptions/medium combination.

Find household relationship between customers

But customers can make spelling mistakes, change order of information in address

Customer changing household information shall change the household relationships

company should be able to send communication at designated frequency, and at required medium to required customer.

Company has 10 days to stop sending information to opted out subscribers.

Company needs a way to figure out if they have sent comms to any customer beyond 10 day opt out period.

Such customers need to be sent in a report. report shall contain: name, communication id, communication medium, opt out date, sending date

Company needs automated statistics, visualizations of sent communications/customer dimensions.

This system shall be secure.

Design shall be scalable.

**Involves**

History processing

Logging/Auditing

Error handling/not sending real error to user

Surrogate key generator