Software Engineering Learnings:

- ## Programming Language : Python
  1. Object oriented(allows, abstraction, Inheritance, Encapsulation).
  2. Having various modules to implement High level functionalities.
  3. Easy to code.
  4. Easy to understand.

- ## Methodology Used :- Agile

  Agile methodology helps improve quality of the product by breaking down the project into manageable units and the project team can focus on high-quality development, testing, and collaboration. This methodology helped us to improve the quality of the product by finding and fixing defects quickly and identifying expectation mismatches early

- ## SOLID Principles:

  ### S — Single responsibility principle :

  In programming, this states that every module or class should have responsibility over a single part of the functionality provided by the software.

  Example from the Project :- Get_face_vector, which is the module only devoted to one task which is getting face vector for the server to recognize the faces, there is no other work being done.

  ### O — Open/closed principle :

  In programming, this states that software entities should be open   for   extensions, but closed for modifications.
  Example from the projects :- Everything is in modules, So every module is open for extension all be needed to make the new utilities and add them into our modules. We don’t need to do anything from scratch.

I — Interface segregation principle

In programming, this states that no client should be forced to depend on methods it does not use. Simply, Do not add additional functionality to an existing interface by adding new methods. Instead, create a new interface and let your class implement multiple interfaces if needed.

Example from Project : All methods in the project, are being used as its fullest and we have classes in all the modules so we can implement any new interface if required.

D - Dependency inversion principle

In programming,this is a way to decouple software modules.
This principle states that :
High-level modules should not depend on low-level modules.
Both should depend on Abstractions.
Abstractions should not depend on details. Details should depend on abstractions.

- ## Clean Code Principles :
  1. Name the variables according to the project domain.
  2. Camel Case for variables and data structures name.
  3. Screaming Snake Case for Constants name.
  4. For methods: Screaming Snake Case for naming them(usually people apply camel case but for more readability we have used this).
  5. Function should only accept 1 or 2 arguments or at max three not more than that.
  6. Followed Indentation, comments(only must needed comments)

- ## Properties of the final Project :-
  1. The project possesses modularity. It is divided into multiple classes and methods.
  2. It is flexible towards expansion
  3. The project is scalable
  4. All aspects of clean code are applied.

- Applied Distribution System : - Ap System

  Availability: A guarantee that every request receives a (non-error) response, without the guarantee that it contains the most recent write.

  Using CAP theorem in this project we have opted for availability over consistency.