**University Of Petroleum And Energy Studies**

**Internship - Low Level Design**

**On**

**Cloud Based Attendance Management System**

Team members:

Akansh Verma

Aviral Mehra

Amish Choudhary

Vedant Sheoran

Guided by: Dr. Gagan Deep

Industry Mentor: Yogesh Ghorpade

**Table of Contents**

1. Introduction...............................................................................................................3
   1. Scope of the document..........................................................................................3
   2. Intended audience.................................................................................................3
   3. System overview..................................................................................................3
2. Low Level System Design.........................................................................................4
   1. Sequence Diagram.............................................................................................4
   2. Navigation Flow/UI Implementation.................................................................5
   3. Screen Validations, Defaults and Attributes......................................................6
   4. Client-Side Validation Implementation.............................................................7
   5. Server-Side Validation Implementation............................................................8
   6. Components Design Implementation................................................................9
   7. Configurations/Settings.....................................................................................9
   8. Interfaces to other components..........................................................................9
3. Data Design................................................................................................................10
   1. List of Key Schemas/Tables in database.......................................................... 10
   2. Details of access levels on key tables in scope................................................. 11
   3. Key design considerations in data design......................................................... 11
4. Details of other frameworks being used.....................................................................12
   1. Session Management......................................................................................... 12
5. Unit Testing...............................................................................................................12

1. Key notes.................................................................................................................. 13
2. Reference.................................................................................................................. 13

**Introduction**

Attendance management system is a software developed for daily student attendance in school, colleges, and institutes. It facilitates access to the attendance information of a particular student in a particular class. The information is stored by the operators, which will be provided by the teacher for a particular class. This system will also help in evaluating attendance eligibility criteria of a student.

**Scope**

The scope of this project is the system on which the software is installed, i.e. the project is developed as a desktop application, and it will work for a particular institute. But later on the project can be modified to operate it online. The following project has much scope both in present as well as future. In the present situation the system can be accessed both in computers and in mobile devices.

In the future the system can be automated using student’s fingerprints.

**System overview**

* Storing the Images of Students in S3 Bucket
* Capturing the image on an Hourly basis
* Loading the image to Face comparison algorithm (compares the faces in
* s3 bucket)
* Marking the attendance for compared faces and storing in DynamoDB.
* Creating a rest API using API gateway and lambda function to connect to dynamo DB through web app
* Creating a web-based dashboard to visualize the attendance

**Low level system Design**

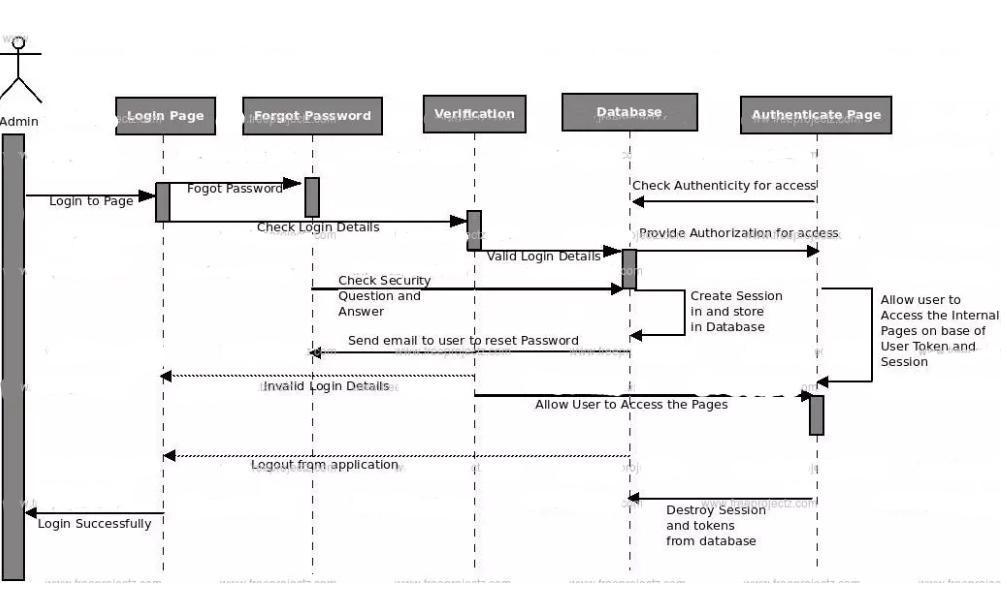
Sequence Diagram

This is the sequence diagram of the attendance management system which shows the interaction between the object of login, attendance, leave, student attendance, employee attendance. The instance of class objects involved in this UML sequence diagram are as follows:

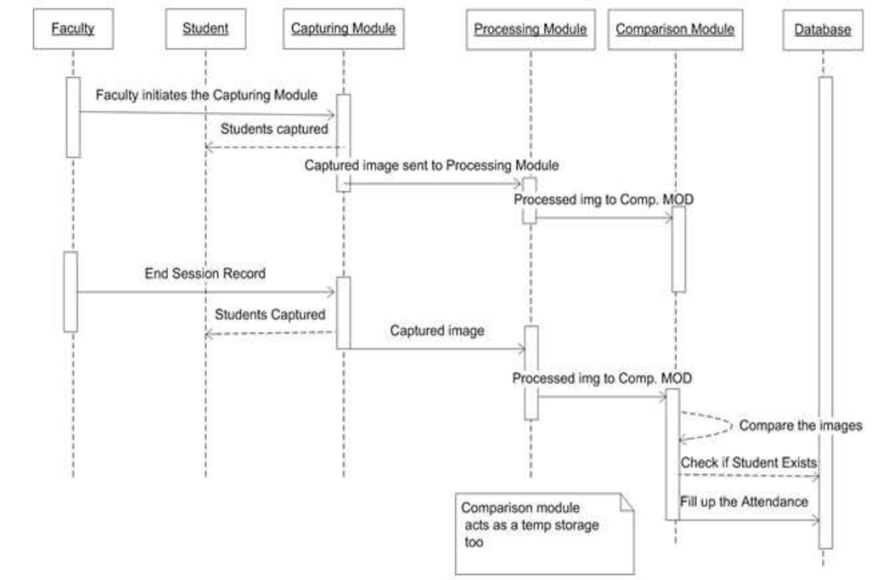
* Login object
* Attendance object
* Leave object
* Student attendance object

Login Sequence

This is the login sequence of the attendance management system, where admin will be able to login in their account using their credentials. After login users can manage all the operations on leave, login, attendance, student attendance. All the pages such as attendance, student attendance are secure and users can access them after login. The diagram below helps demonstrate the login system of the attendance management system.



Sequence after successful login



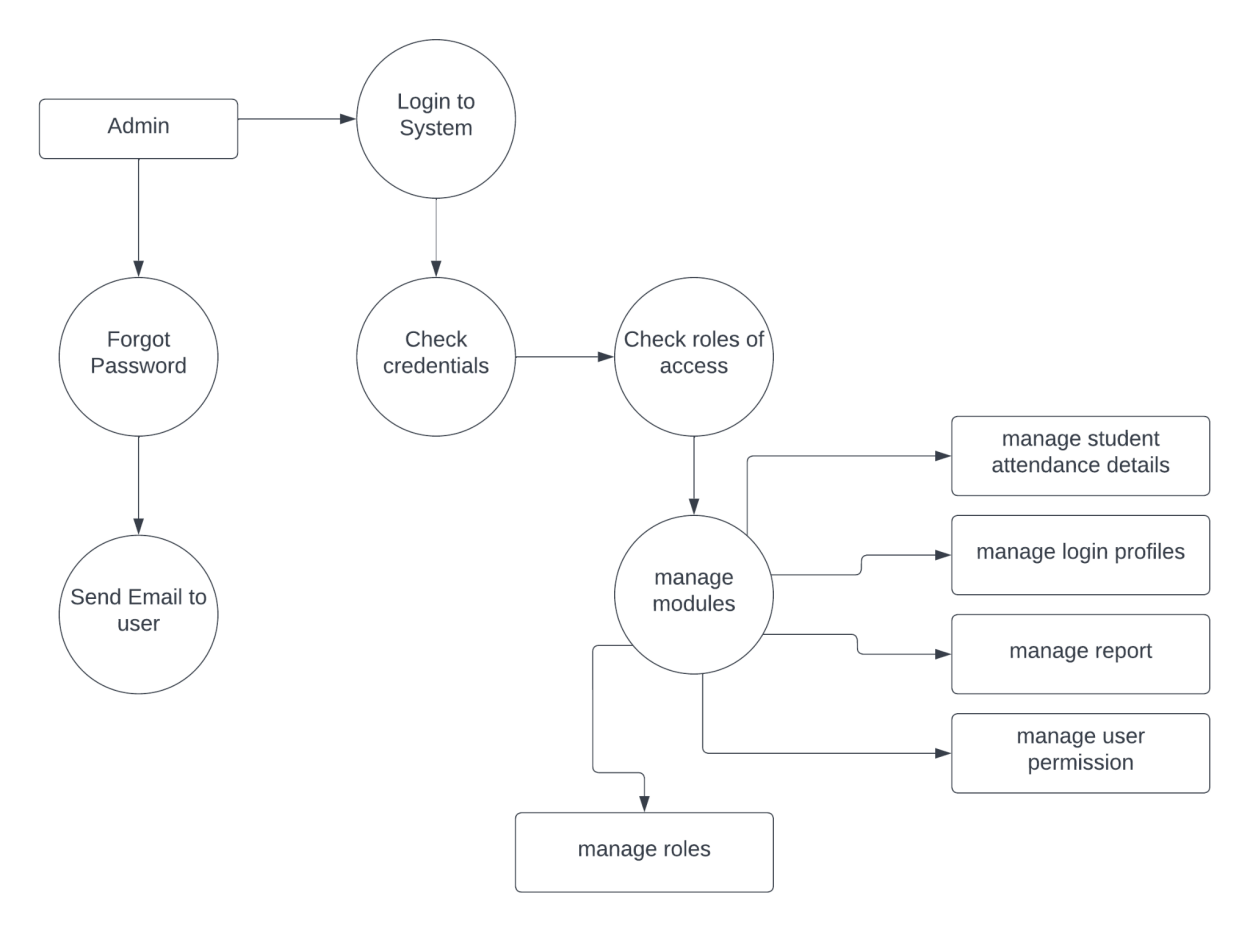
Data flow diagram (2nd level)

DFD level 2 goes 1 step deeper into the parts of level 1 of attendance. It may require more functionalities of attendance to reach the necessary level of detail about the attendance functioning. First level DFD of the attendance management system shows how the system is divided into subsystems(processes). The second level DFD contains more details of ##keyword7##, login, leave, holidays, student attendance, attendance.

Low level functionalities of the attendance system-

* Admin login to the system and manage all the functionalities of the attendance system.
* Admin can add, edit, delete and view the records of attendance, student attendance, leave.
* Admin can manage all the details of attendance, holidays, login.
* Admin can also generate reports of attendance, student attendance, holidays, leave.
* Admin can search the details of student attendance, leave, login.
* Admin can apply different levels of filters on reports of attendance, holidays, leave.
* Admin can track the detailed information of student attendance, holidays, leave.

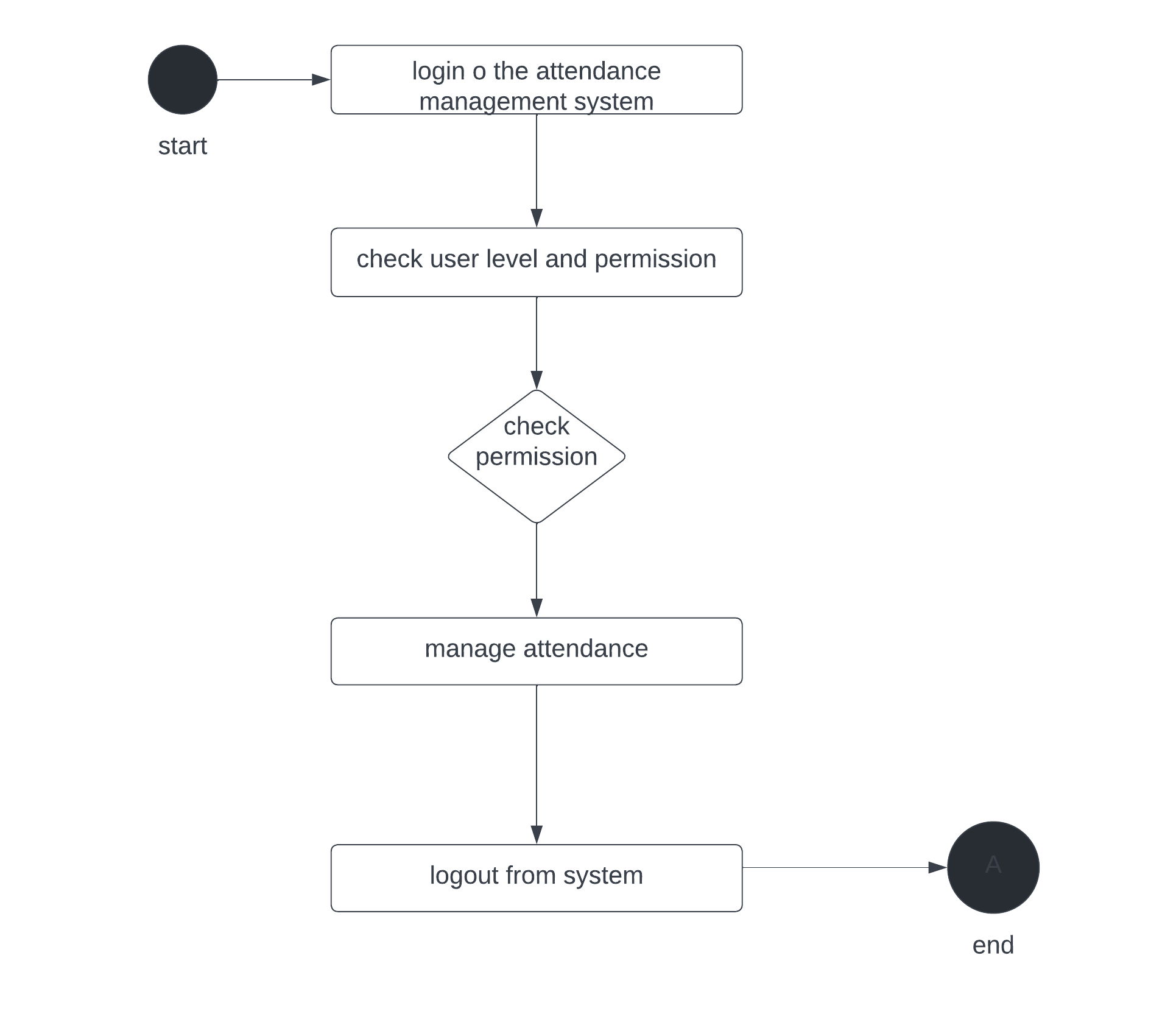
DFD level 2



Activity diagram

This is the activity UML diagram of the attendance management system which shows the flow between the activity of leave, login, attendance, holidays. The main activity involved in this activity diagram are as follows:

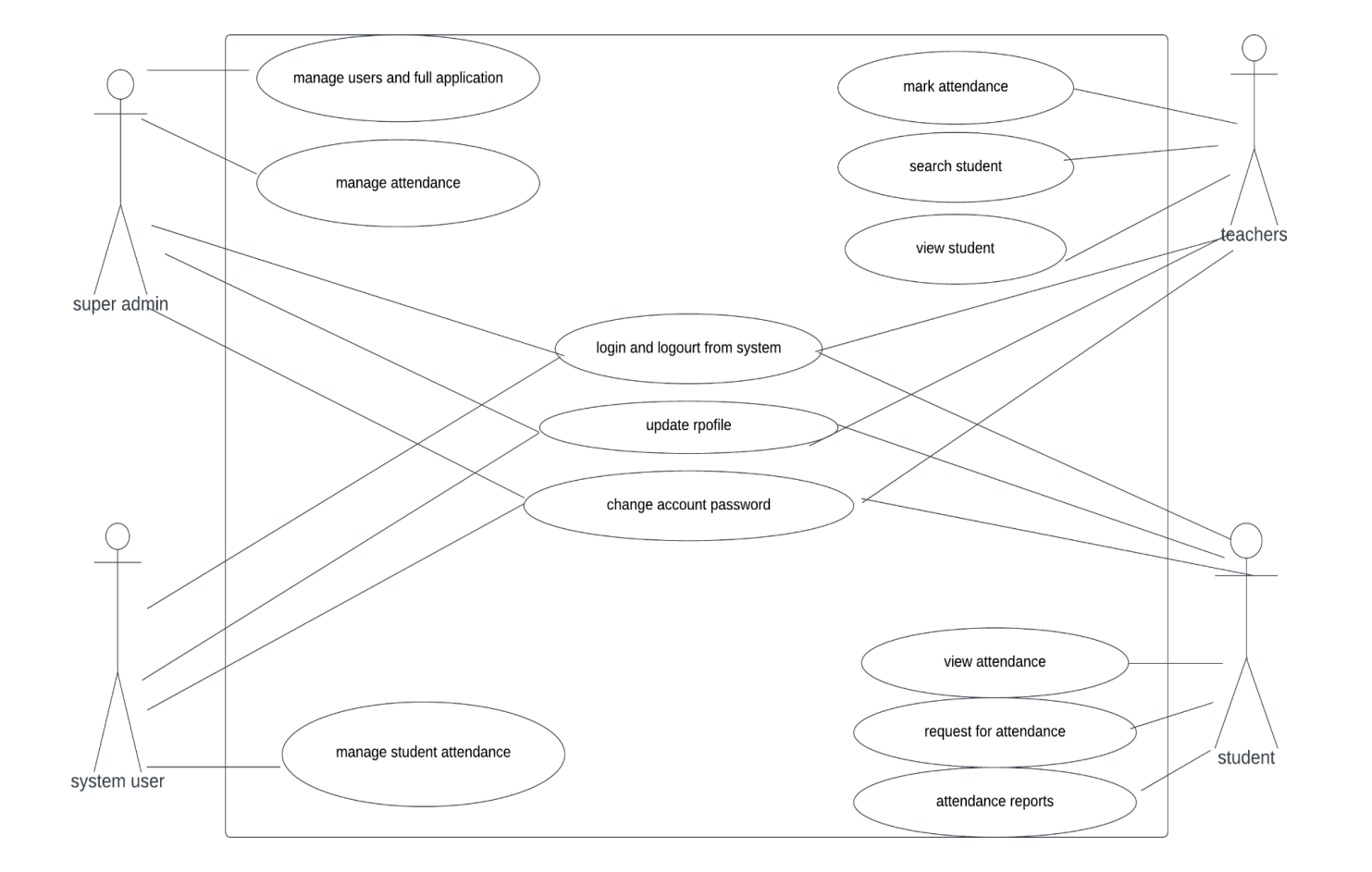
* Attendance activity
* Login activity



Use case Diagram

This use case diagram is the graphical depiction of the interaction among the elements of the attendance management system. It represents the methodology used in the system analysis to identify, clarify and organize the system requirements of the attendance system. The main actors of the attendance system in this use case diagram are: super admin, system user, teachers, student, who perform the different type of use cases such as manage attendance, manage holidays, manage leave, manage login, manage system users, manage users, and full attendance management system operations. Major elements of the use case diagram are as follows:

* Super admin entity: Use cases of super admin are manage attendance, manage holidays, manage leave, manage login, manage system user, manage users.
* System user entity: Use cases of system user are manage attendance, manage holidays, manage leave.
* Teacher entity: Use cases of teachers are mark attendance, search students, view student profiles.
* Student entity: Use cases of students are view attendance, Request for attendance, view attendance reports.



Component Diagram

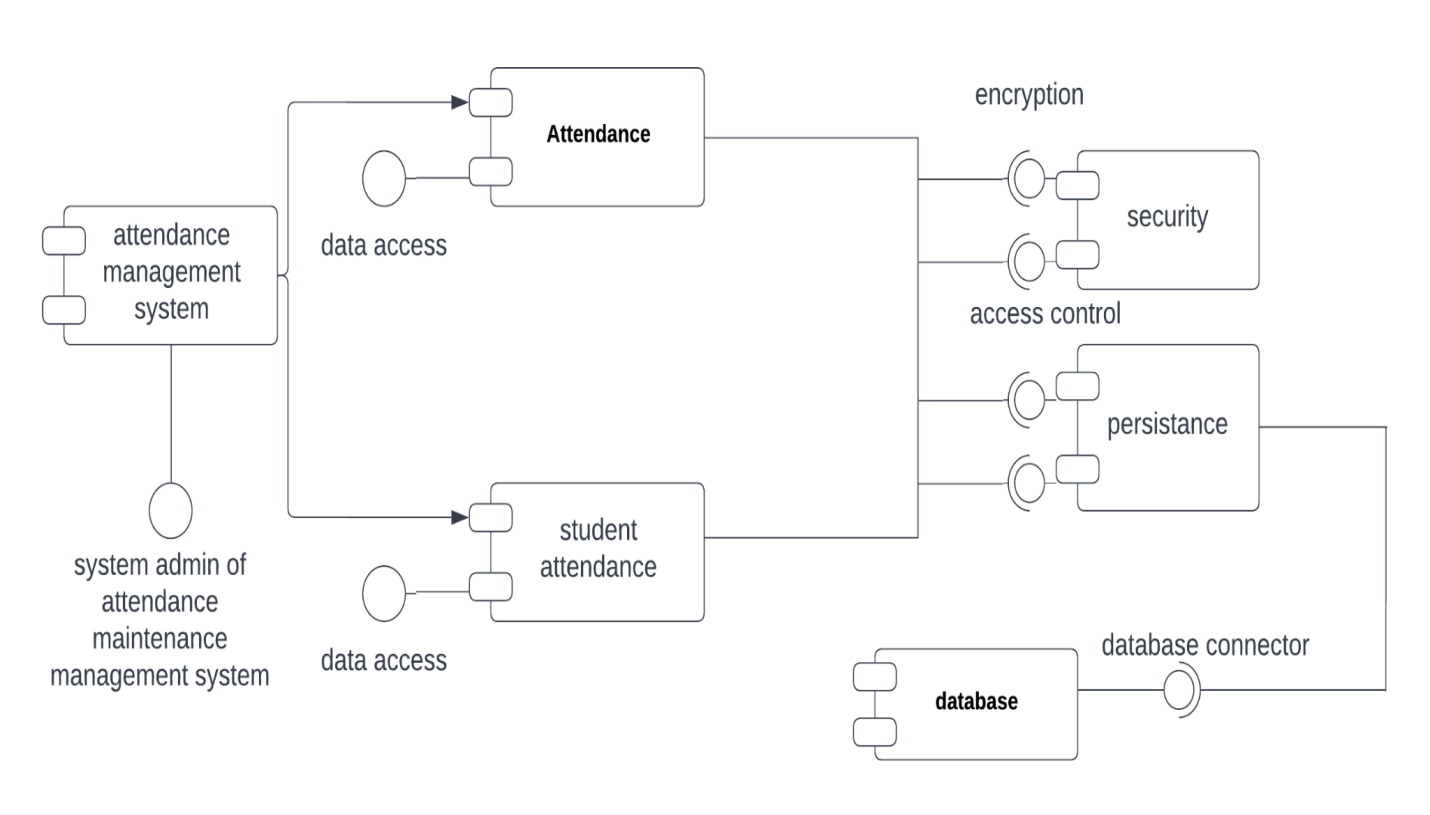
This is a component diagram of the Attendance management system which shows components, provided and required interfaces, ports and relationships between the attendance, login, leave. These types of diagrams are used in component-based development to describe systems with Service oriented architecture. Attendance management system component diagram describes the organization and wiring of the physical components in a system.

Components of UML component diagram:

* Attendance component
* login component
* leave component

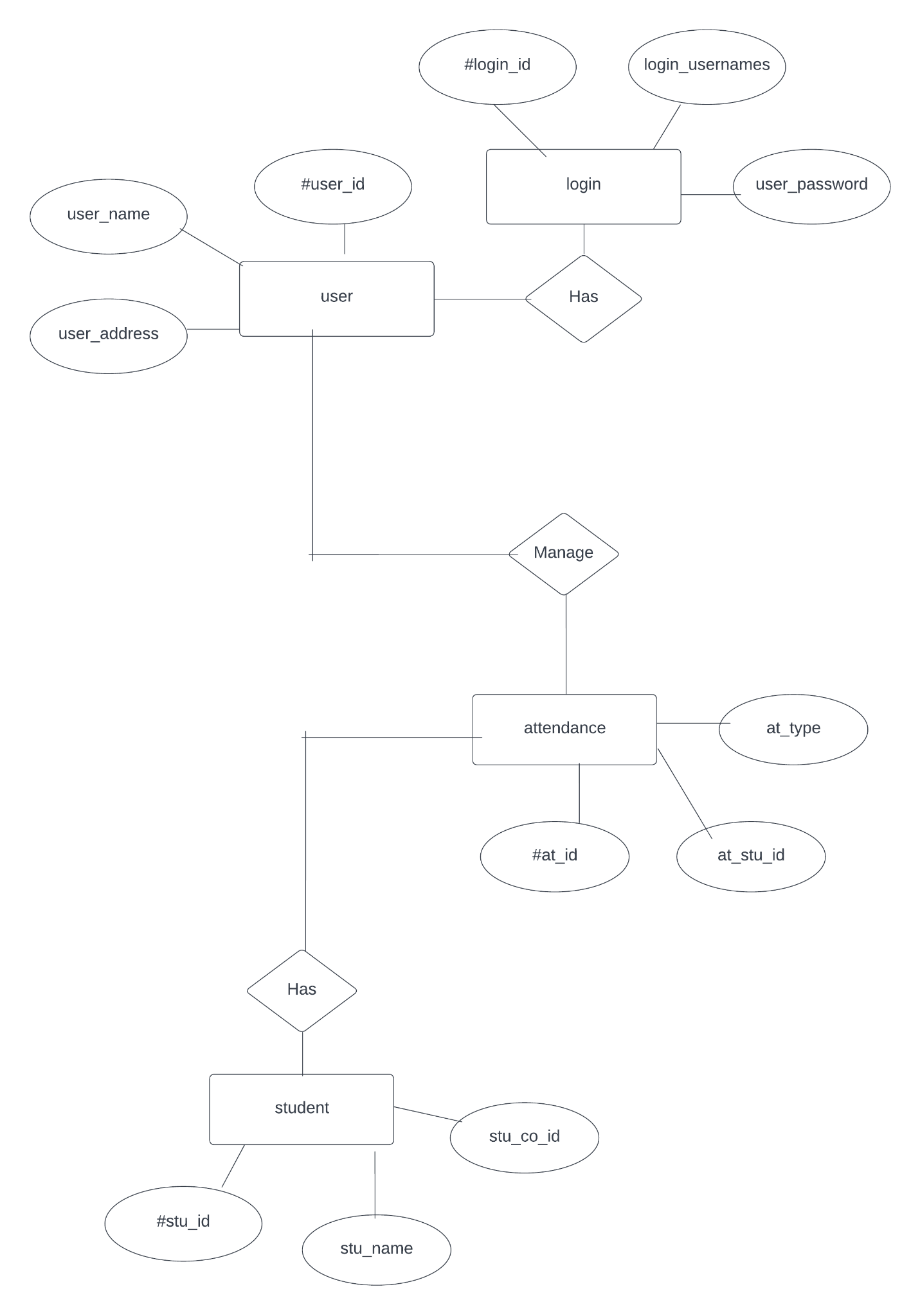
Features of attendance management system component diagram:

* You can show the models the components of the attendance management system.
* Model the database schema of the attendance management system.
* Model the executables of an application of the attendance management system.
* Model the system’s source code of attendance management system.



**Data Design**

Entity-Relationship diagram of the attendance management system



This ER diagram represents the model of the attendance system entity. The ER diagram of the attendance management system shows all the visual instrument of database tables and the relationship between students, holidays, login, etc. It used structure data and to define the relationships between structure data groups of attendance management system functionalities. The main entities of the attendance management system are Attendance, Student, holiday, leave and login.

Attendance management system entities and their attributes:

* Attendance entity: Attributes of attendance are attendance\_id, attendance\_student\_id, attendance\_type, attendance\_description.
* Employee entity: Attributes of employee entity are employee\_id, employee\_name, employee\_name, employee\_mpbile, employee\_email, employee\_username, employee\_passd, employee\_address.
* Student\_entity: Attributes of student are student\_id, tudent\_college\_id, student\_name, student\_mobile, student\_email, student\_username, student\_password, student\_address.
* Holidays entity: Attributes of holidays are holidays\_id, holidays\_employee\_id, holidays\_date, holidays\_dusration, holidays\_description.
* Leave entity: Attribute of leave entity are leave\_id, leave\_employee\_id, leave\_type, leave\_status, leave\_to, leave\_from, leave\_description.
* login entity: attribute of login are login\_id, login\_user\_id, login\_role\_id, login\_username, login\_password, login\_lastlogin.

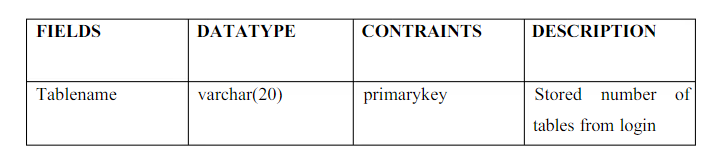
Description of the attendance management system database:

* The details of attendance is store into the attendance tables respective with all the tables.
* Each entity(login, student, leave, employee, attendance) contains primary key and unique keys.
* The entity student, leave has binded with attendance, employee entities with foreign key.
* There is one-to-one and one-to-many relationships available between leave, holidays, login, attendance,
* All the entities attendance, leave, student, login are normalized and reduce duplicacy of records.
* We have implemented indexing on each tables of attendance management system tables for fast query execution.

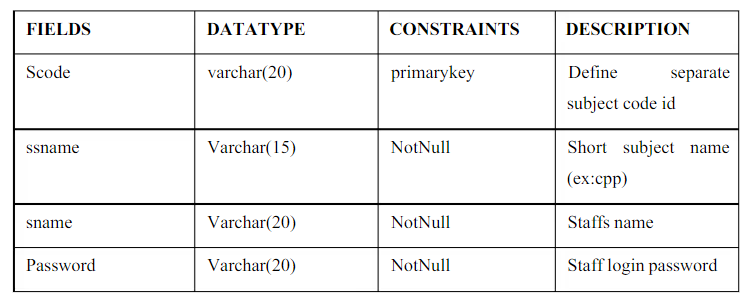
Data Design

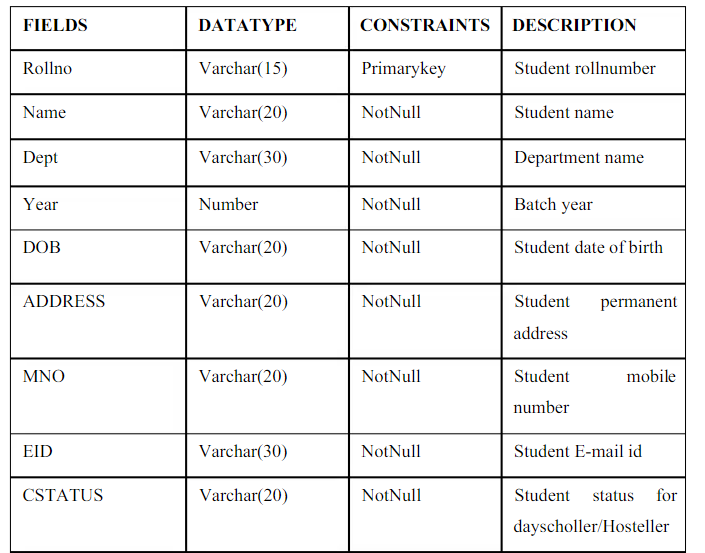
List of Key Schemas/Tables in database

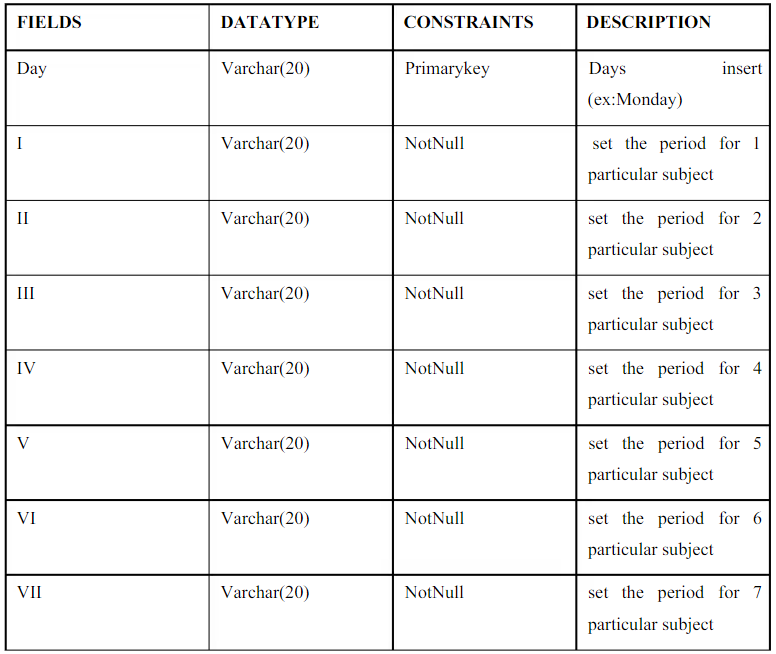
* Login Table



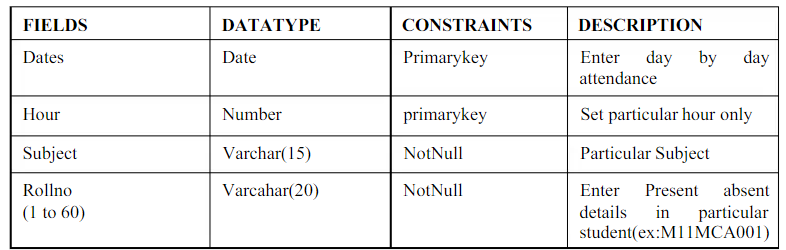
* Teacher’s Table



* Student’s Table
* Time Table



* Attendance Table



**Frameworks used**

Python Web Frameworks:

A Web framework is a collection of packages or modules which allow developers to write Web applications (see ​WebApplications​) or services without having to handle such low-level details as protocols, sockets or process/thread management​. ​A web application may use a combination of a base HTTP application server, a storage mechanism such as a database, a template engine, a request dispatcher, an authentication module and an AJAX toolkit. These can be individual components or be provided together in a high-level framework. ​The most popular high-level frameworks are ​Django​,​TurboGears​,​web2py​ etc.

AWS DynamoDB:

Amazon DynamoDB is a NoSQL database that supports key-value and document data models, and enables developers to build modern, serverless applications that can start small and scale globally to support petabytes of data and tens of millions of read and write requests per second. DynamoDB is designed to run high-performance, internet-scale applications that would overburden traditional relational databases.Amazon DynamoDB is a key-value and document database that delivers single-digit millisecond performance at any scale. It's a fully managed, multiregion, multimaster, durable database with built-in security, backup and restore, and in-memory caching for internet-scale applications

AWS Rekognition:

Amazon Rekognition makes it easy to add image and video analysis to your applications using proven, highly scalable, deep learning technology that requires no machine learning expertise to use. With Amazon Rekognition, you can identify objects, people, text, scenes, and activities in images and videos, as well as detect any inappropriate content. Amazon Rekognition also provides highly accurate facial analysis and facial search capabilities that you can use to detect, analyze, and compare faces for a wide variety of user verification, people counting, and public safety use cases.

Amazon S3:

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. This means customers of all sizes and industries can use it to store and protect any amount of data for a range of use cases, such as websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. Amazon S3 provides easy-to-use management features so you can organize your data and configure finely-tuned access controls to meet your specific business, organizational, and compliance requirements.

AWS API Gateway:

Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. APIs act as the "front door" for applications to access data, business logic, or functionality from your backend services. Using API Gateway, you can create RESTful APIs and WebSocket APIs that enable real-time two-way communication applications. API Gateway supports containerized and serverless workloads, as well as web applications.

API Gateway handles all the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls, including traffic management, CORS support, authorization and access control, throttling, monitoring, and API version management. API Gateway has no minimum fees or startup costs. You pay for the API calls you receive and the amount ​of data transferred out and, with the API Gateway tiered pricing model, you can reduce your cost as your API usage scales.

AWS Lambda:

AWS Lambda lets you run code without provisioning or managing servers. With Lambda, you can run code for virtually any type of application or backend service - all with zero administration. Just upload your code and Lambda takes care of everything required to run and scale your code with high availability. You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app.

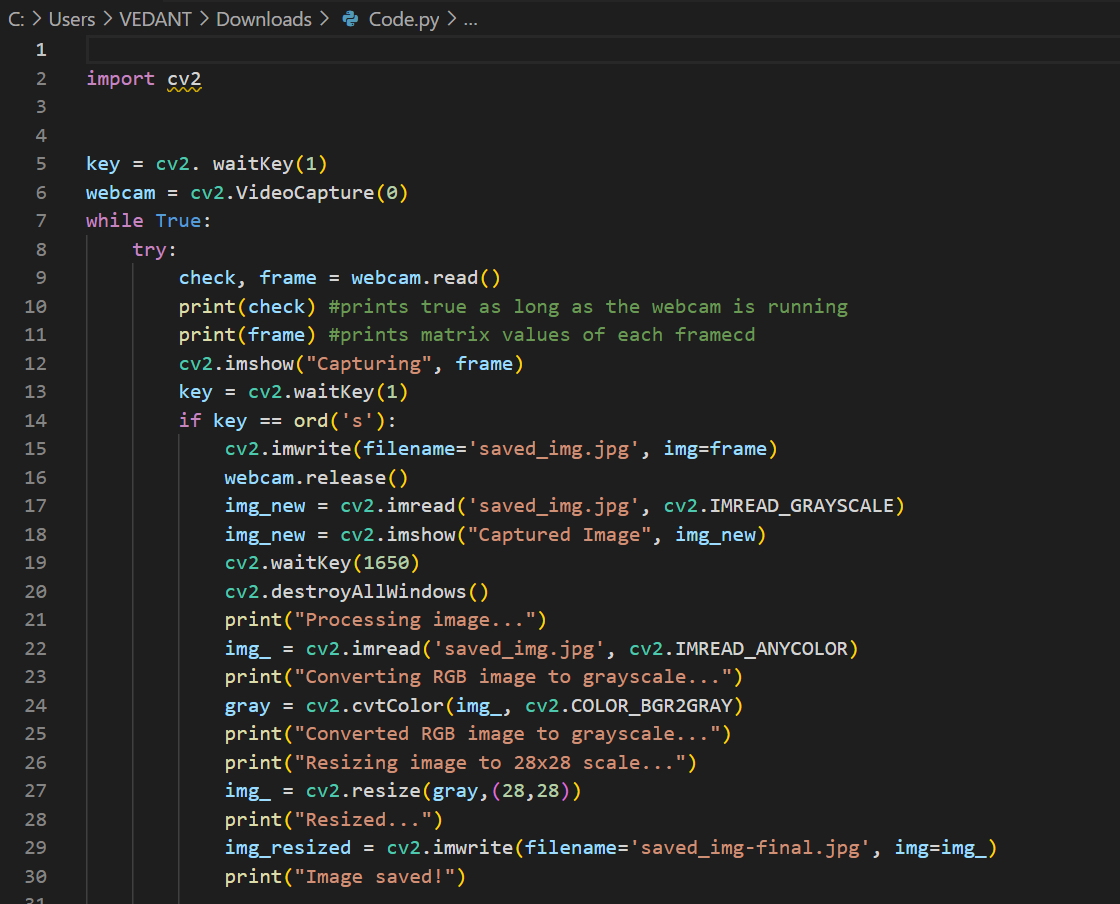
Session Management

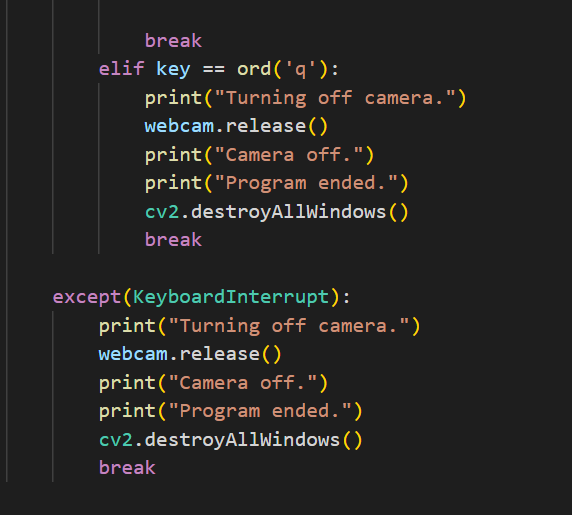
Current business scenario demands high scalability & availability of the hosted solution, but this approach has limitations of horizontal scaling for large scale system requirements. AWS cloud platform has all the required infrastructure, network components designed for horizontal scaling across multiple zones to ensure high availability on demand to make most efficient use of resources and minimize the cost. Application deployment / migration on AWS requires pro-active thinking especially in the case of application session management.

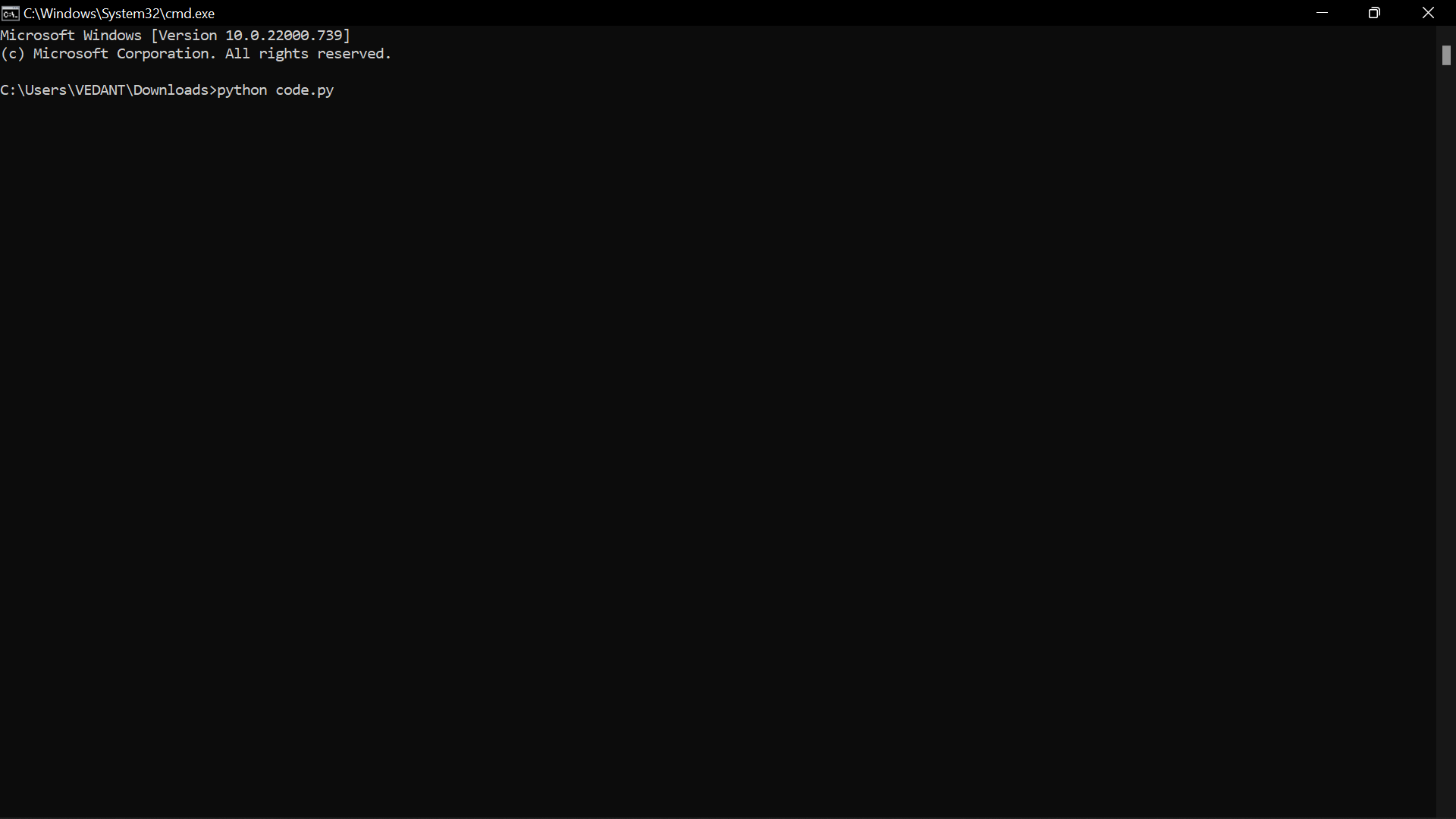
Considering our scenario mentioned above, the developer enables server sessions. When our application needs scale up (Horizontal Scaling) from one to many servers, we will deploy the application server behind a load balancer. By default Elastic load balancer routes each user’s request to the application instance with less load using round robin algorithm. We have to ensure that load balancer sends all requests from a single user to the same server where is session is created. In this scenario, ELB sticky session (also known as session affinity) comes in handy as it does NOT require any code changes within the application. When we enable sticky session in ELB, the ELB keeps track of all user requests and which server it has routed their past requests and start sending requests to the same server.

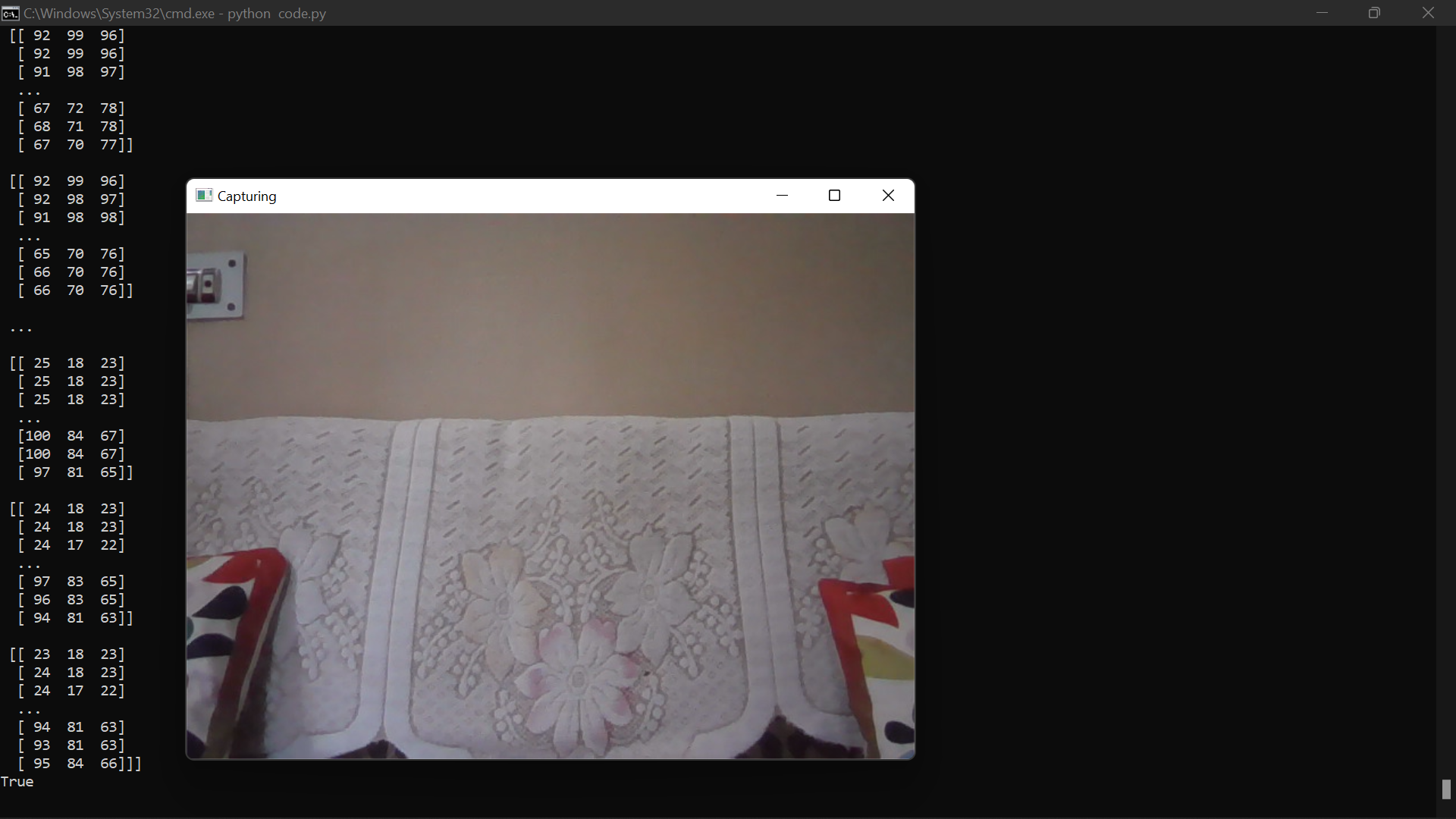
The challenges faced while using RDBMS for Session storing was its administration workload as well as scalability. AWS Dynamo DB is a NoSQL database that can handle massive concurrent read and writes. Using AWS Dynamo DB console one can configure reads/writes per second and accordingly Dynamo DB will provision the required infrastructure at the backend. So scalability and administration needs are taken care by the service itself. Internally all data items are stored on Solid State Drives (SSDs) and are automatically replicated across three Availability Zones in a Region to provide built-in high availability and data durability.

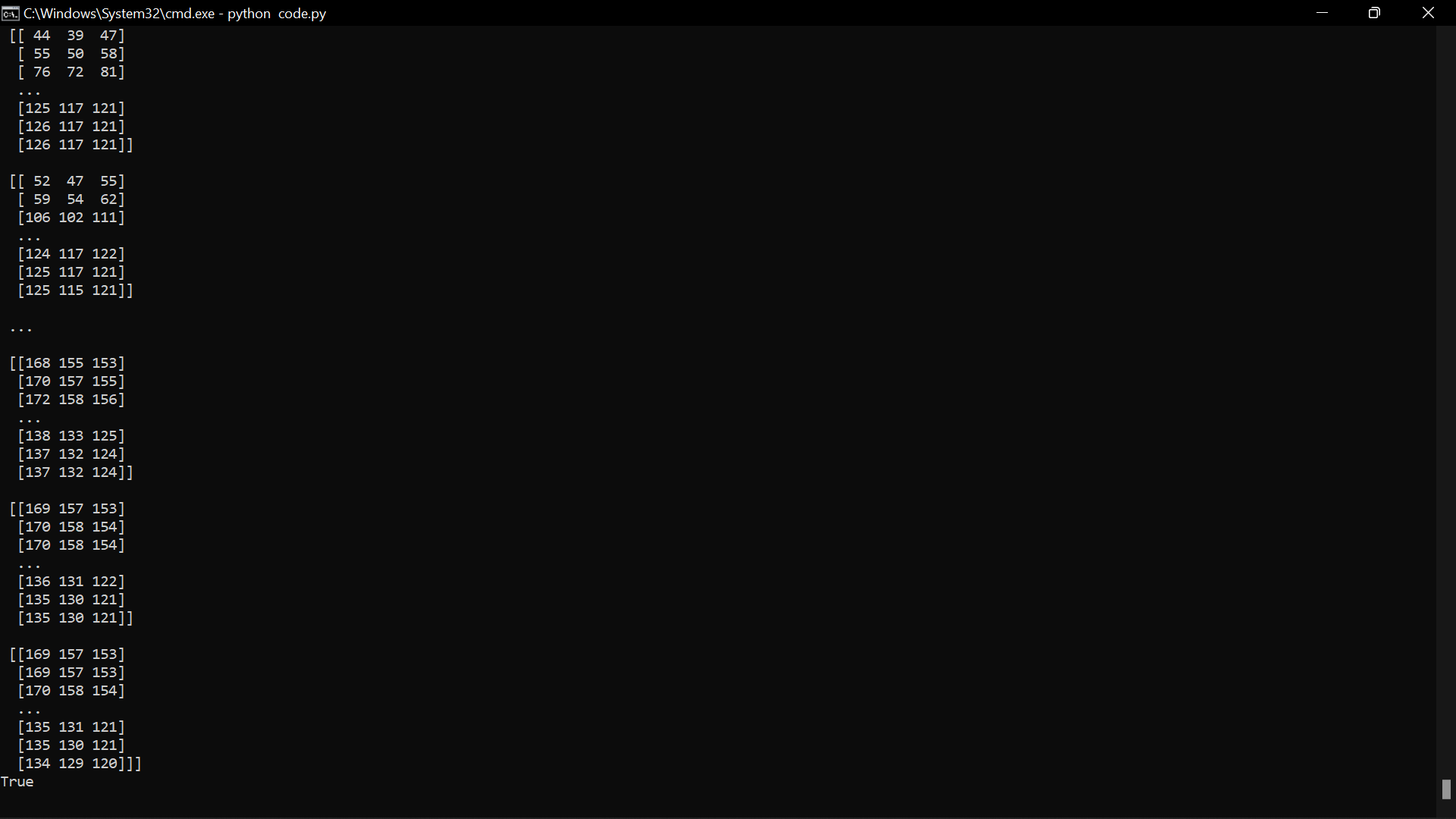
Unit Testing

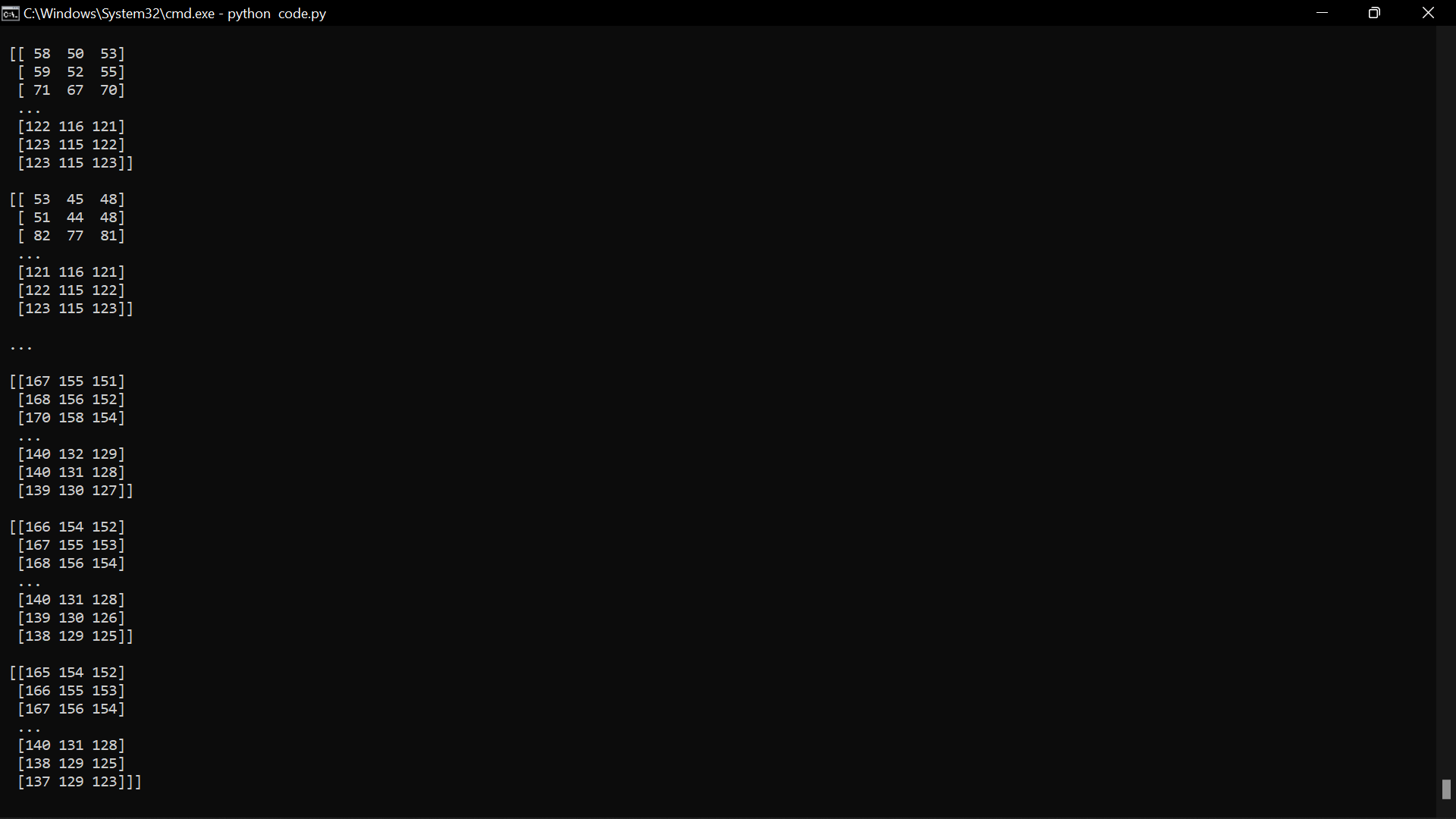












On pressing S image is captured and saved



Captured image :



Saved Image after resizing into the given scale

