PROJECT REPORT
ON

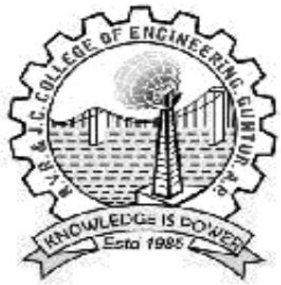
FACE RECOGNITION BASED ATTENDANCE MANAGEMENT SYSTEM (FRAMS)

Submitted in partial fulfillment of requirements to

IT 362 – PROJECT I

By

A. JAYA MADHURI (Y18IT001)
D.NIMSY (Y18IT018)
M.BHAVYA (Y18IT055)



APRIL 2021

R.V.R & J.C.COLLEGE OF ENGINEERING(AUTONOMOUS)
(NAAC A+ GRADE) (Approved by A.I.C.T.E)
(Affiliated to Acharya Nagarjuna University)
Chandramoulipuram : : Chowdavaram
GUNTUR – 522 019

R.V.R & J.C.COLLEGE OF ENGINEERING

DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this project work titled FACERECOGNITION BASED ATTENDANCE MANAGEMENT SYSTEM is the bonafide work of (A.Jaya Madhuri(Y18IT001), D. Nimsy (Y18IT018), M. Bhavya(Y18IT055)) who have carried out the work under my supervision, and submitted in partial fulfillment of the requirements to **IT-362, PROJECT I** during the year 2020-2021.

Dr. M. Pompapathi
Lecturer Incharge

Dr. A.Srikrishna
Prof.&HOD, Dept. of IT

ACKNOWLEDGEMENTS

The successful completion of any task would be incomplete without a proper suggestions, guidance and environment. Combination of these three factors acts like backbone to our Project **"FACE RECOGNITION BASED ATTENDANCE MANAGEMENT SYSTEM".**

We are very much thankful to **Dr.A.SUDHAKAR**, Principal of R.V.R. & J.C. College of Engineering, Guntur for having allowed delivering this Project - I.

We express our sincere thanks to **Dr.A.Srikrishna**, Professor, Head of the Department of Information Technology for her encouragement and support to carry out this mini project successfully.

We are very glad to express our special thanks to **Dr.M.Pompapathi** , Associate Professor in Department of Information Technology for timely, guidance and providing us with most essential materials required for the completion of this report.

Finally we express our sincere thanks to all the **Teaching** and **Non-Teaching staff** of **IT Department** who have contributed for the successful completion of this report.

A.JAYA MADHURI (Y18IT001)
D.NIMSY(Y18IT018)
M.BHAVYA(Y18IT055)

CONTENTS

1. PROBLEM STATEMENT

FACE RECOGNITION BASED ATTENDANCE MANAGEMENT SYSTEM

The most accurate way to maintain an attendance management system is through 'Face Recognition' .The proposed system facilitates users to view their detailed attendance report and helps to maintain their attendance.

Now a days, there are many technologies such as biometric, RFID and speech recognition management system. But biometric attendance management system uses finger print sensor , in this system there were lot of problems that were being faced in the crowded places, as the number of devices were limited and number of people were more and a single person can use the system at a time to record attendance so it takes more time to record attendance of whole strength. The concept of RFID was simple to provide a chip inside an ID card, but the system required large number of components and space. Though the chance of error in this system was very less, but the equipment cost was high and the maintenance was difficult. Further speech recognition system, was identified in this system the speech was taken and the attendance was recorded. The system is bit complicated as there was no assurance thing, anyone could mark for anyone by recording their voice on the phone and playing it, so the system was very unreliable.

The proposed attendance management system uses the concept of Face Recognition. The proposed system facilitates

- User needs to register into the system through their user name or email id.
- During registration user just needs to upload a photo to complete the registration process.
- After registering, user needs to log in to the system to view his/her detailed attendance report.
- The registered users needs to appear in front of the camera just for 1 second to record his/her attendance into the system.
- It also facilitates admins ,who are responsible for starting the attendance marking into the system automatically according to the specified amount of time.
- It also facilitates the admins to view the detailed attendance report of registered users just by logging into the system.

In this system user needs to appear infront of webcam even just for 1 second to record attendance. This attendance management system provides user to register through their user name or email id and just need to upload a photo for face detection and recognition while marking attendance. After registering , user can be able to view his detailed attendance report and percentage of attendance up to the date . Our system also contains admins ,who are responsible for starting the attendance marking into the system automatically according to their time which is already specified . And admin can also view the detailed attendance sheet of each user just by logging into the system.

2. SRS DOCUMENTATION - REQUIREMENTS ELICITATION

ID	Details	Functionalities	Priorities
R1	FRAMS must be able to store user information	Functional Data	Must have
R2	FRAMS must be able to store image of the user	Functional Data	Must have
R3	FRAMS must be able to respond to user within seconds	Non-Functional performance	Must have
R4	FRAMS must be able to validate login credentials of user	Functional	Must have
R5	FRAMS must be able to generate attendance report for each user	Functional	Must have
R6	FRAMS must be able to display attendance report correctly as the user logs in	Functional	Must have
R7	FRAMS must be able to display basic details of corresponding user when user logs in	Functional	Must have
R8	FRAMS must be able to calculate attendance percentage for each user	Functional	Must have
R9	FRAMS must be able to display attendance percentage corresponding user who logged in	Functional	Must have
R10	FRAMS must be able to respond to the user and display attendance report within 5 seconds	Non-Functional	Must have
R11	FRAMS must be able to redirect to login page after registering in the system	Non-Functional	Must have
R12	FRAMS must be able to logout of the system	Functional	Must have
R13	FRAMS must be able to provide user registration form	Functional	Must have
R14	FRAMS must be able to provide user login form	Functional	Must have
R15	FRAMS must be able to provide admin login form	Functional	Must have
R17	FRAMS must be able to store admin login information	Functional	Must have
R18	FRAMS must be able to respond to admin within 5 seconds	Non-Functional	Must have

R19	FRAMS must be able to display respective admin basic information after logging in	Functional	Must have
R20	FRAMS must be able to generate users detailed attendance report when requested by admin	Functional	Must have
R21	FRAMS must be able to logout of the system	Functional	Must have
R22	FRAMS must be able to start the attendance marking	Non-Functional	Must have
R23	FRAMS must be able to stop the attendance marking when required	Non-Functional	Must have
R24	FRAMS must be able to detect all the faces in current frame	Functional	Must have
R25	FRAMS must be able to match the faces in current frame with images in the database system	Functional	Must have
R26	FRAMS must be able to recognize all the faces in current frame	Functional	Must have
R27	FRAMS must be able to recognize the faces within 3 seconds	Non-Functional	Must have
R28	FRAMS must be able to display the name of use while face recognition	Non-Functional	Must have
R29	FRAMS must be able to update the attendance report after recognizing the face	Functional	Must have
R30	FRAMS must be able to update the attendance percentage each time after marking attendance of each user	Functional	Must have
R31	FRAMS must be able to update attendance percentage for absent users	Functional	Must have
R32	FRAMS must be able to accept attendance accurately for specified amount of time requested by admin	Functional	Must have
R33	FRAMS must be able to generate attendance report according to dates specified by user	Functional	Must have
R34	FRAMS must be able to validate the start date and end date specified by the user	Functional	Must have
R35	FRAMS must be able to generate the attendance report according to dates specified by admin for any user	Functional	Must have

R36	FRAMS must be able to validate start date and end date specified by admin	Functional	Must have
R37	FRAMS must be able to change the password of user when requested	Functional	Must have
R38	FRAMS must be able to change password of admin when requested	Functional	Must have
R39	FRAMS must be able to update profile of user when required	Functional	Must have
R40	FRAMS must be able to update admins profile when required	Functional	Must have

3. SYSTEM REQUIREMENT SPECIFICATION

Software Requirements:

- Operating System : windows XP
- Coding language : PHP , HTML , PYTHON
- Data Base : MySQL

Hardware Requirements:

- Personal computer with keyboard and mouse maintained with uninterrupted power supply.
- Processor : Intel® core™ i5
- Installed Memory (RAM) : 1.00 GB
- Hard Disc : 40 GB

4. REQUIREMENTS MODELING

The most important factor for the success of an IS project is whether the software product satisfies its users' requirements. Models constructed from an analysis perspective focuses on determining these requirements. This means Requirement Model includes gathering and documenting facts and requests.

The use case model gives a perspective on many user requirements and models them in terms of what the software system can do for the user. Before the design of software which satisfies user requirements, we must analyze both the logical structure of the problem situation, and also the ways that its logical elements interact with each other. We must also need to verify the way in which different, possibly conflicting, requirements affect each other. Then we must communicate this under standing clearly and unambiguously to those who will design and build the software.

Use-case diagrams graphically represents system behavior (use cases). These diagrams present a high level view of how the system is used as viewed from an outsider's (actor's) perspective. A use-case diagram may contain all or some of the use cases of a system.

A use-case diagram can contain:

- ·actors ("things" outside the system)
- ·use cases (system boundaries identifying what the system should do)
- interactions or relationships between actors and use cases in the system including the associations, dependencies, and generalizations.

Use-case diagrams can be used during analysis to capture the system requirements and to understand how the system should work. During the design phase, you can use use-case diagrams to specify the behavior of the system as implemented.

5. IDENTIFICATION OF ACTORS and USECASES

Identification of ACTORS :

Actors represent system users. They are NOT part of the system .They represent anyone or anything that interacts with the system.
An actor is someone or something that:

- Interacts with or uses the system
- Provides input to and receives information from the system
- Is external to the system and has no control over the use cases

Actors are discovered by examining:

- Who directly uses the system
- Who is responsible for maintaining the system
- External hardware used by the system
- Other systems that need to interact with the system

The needs of the actor are used to develop use cases. This insures that the system will be what the user expected.

Graphical depiction:

An actor is a stereotype of a class and is depicted as a "stickman" on a use-case diagram. For example,

Student

Actors identified in the information system are:

1) Admin
2) User

1) Admin: Admin has to login into his account
    - to view his/her profile,
    - to start the attendance marking,
    - to view users attendance report,
    - to update his/her profile,
    - to change his/her password request

   and he has to logout the account after the desired actions complete.

2) User: User has to login into his account
   - to view his/her profile,
   - to view his/her attendance report,
   - to update his/her profile,
   - to change his/her password request

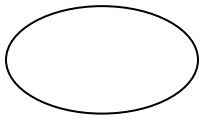
and he has to logout the account after the desired actions complete.

Identification of Use-Cases Or Sub Use-Cases

Use case can be described as a specific way of using the system from a user's perspective. A more detailed description might characterize a use case as:

- A pattern of behavior the system exhibits
- A sequence of related transactions performed by an actor and the system

The UML notation for use case is:



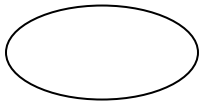
Purpose of usecases:

- Well structured use cases denote essential system or subsystem behaviours only, and are neither overly general nor too specific.
- A use case represents a functional requirement of the system as a whole
- Use cases represent an external view of the system
- A use case describes a set of sequences, in which each sequence represents the interaction of the things outside the system with the system itself.

Use-cases identified for face recognition based attendance management system are:

1 .**Use-case name:** LOGIN

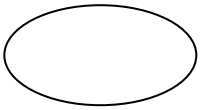
This is a use case which is used by actor to log on to the system and view the available set of operations that he can perform



Login

2.**Use-case name :** VIEW PROFILE

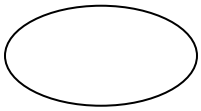
System allows user or admin to view his/her profile and can be able to select available functionalities respectively.



View profile

3.**Use-case name:** UPDATE PROFILE

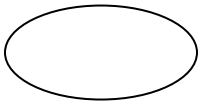
This use case allows user or admin to update his/her profile like updating existing email id, phone number, date of birth etc.,



Update profile

4.**Use-case name:** CHANGE PASSWORD

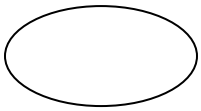
This use case allows the user or admin to change the password and secure delivery contact information.



Change password

5 .**Use-case name:** VIEW USER ATTENDANCE

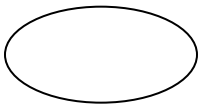
This use case allows admin to view the attendance report of any registered user



View user attendance

6**. Use-case name:** VIEW ATTENDANCE

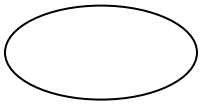
This use case allows the registered to view his /her detailed attendance report or date wise attendance report.



View attendance

7. **Use-case name:** MARK ATTENDANCE

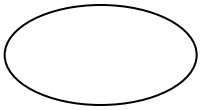
This use case allows admin to start marking attendance for a specified amount of time in two sessions i.e, morning session and afternoon session.



Mark attendance

8 .**Use-case name:** USER REGISTRATION

This usecase allows a person to register as a user into the system by supplying basic details.



User registration

Identification of RELATIONS

Association Relationship:

An association provides a pathway for communication. The communication can be between use cases, actors, classes or interfaces. If two objects are usually considered independently, the relationship is an association.

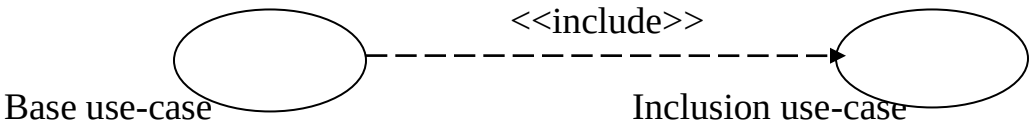
Dependency Relationship:

A dependency is a relationship between two model elements in which a change to one model element will affect the other model element. Use a dependency relationship to connect model elements with the same level of meaning.

We can provide here

1. Include relationship:

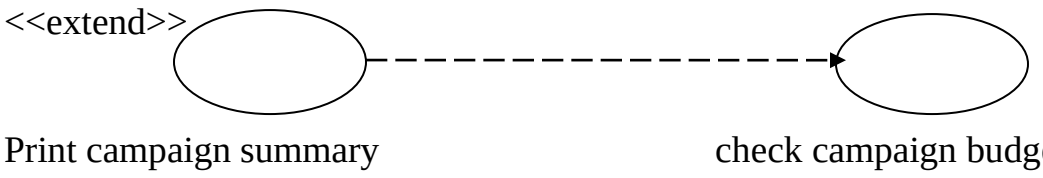
   It is a stereotyped relationship that connects a base use case to an inclusion use case .An include relationship specifies how the behavior in the inclusion use case is used by the base use case.



2. Extend relationship:

   It is a stereotyped relationship that specifies how the functionality of one use case can be inserted into the functionality of another use case.

   <<extend>> is used when you wish to show that a use case provides additional functionality that may be required in another use case.

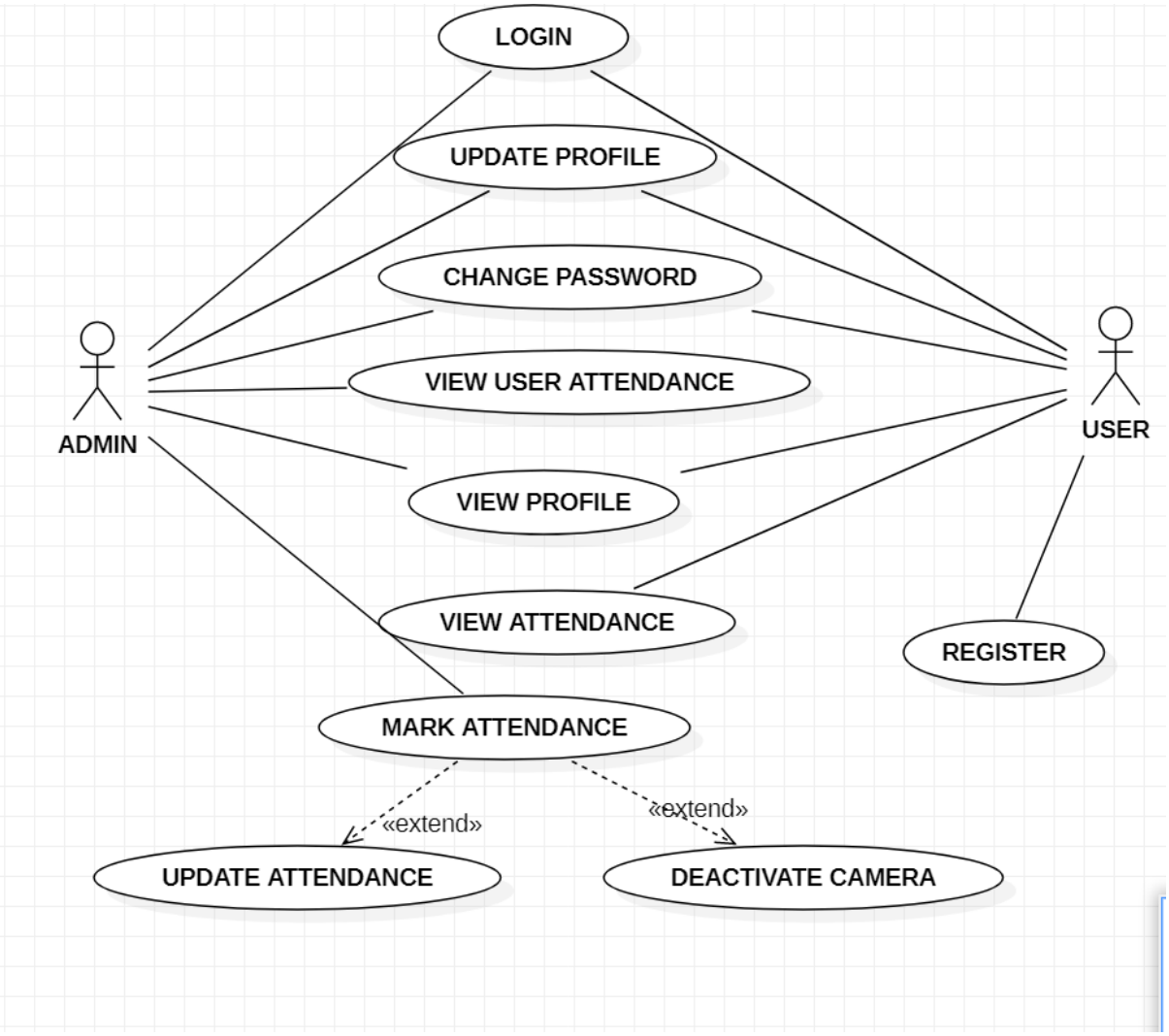


6. CONSTRUCTION OF USE CASE DIAGRAM AND FLOW OF EVENTS.

Use-case diagrams graphically represent system behavior. These diagrams present a high level view of how the system is used as viewed from an outsider's perspective.

Use-case diagrams can be used during analysis to capture the system requirements and to understand how the system should work. During the design phase, you can use use-case diagrams to specify the behavior of the system as implemented.

USE CASE DIAGRAM FOR FRAMS:



FLOW OF EVENTS

A flow of events is a sequence of transactions performed by the system. They typically contain very detailed information .Flow of events document is typically created in the elaboration phase.

Each use case is documented with flow of events

- A description of events needed to accomplish required behaviour
- Written in terms of what the system should do, NOT how it should do it
- Written in the domain language , not in terms of the implementation

A flow of events should include

- When and how the use case starts and ends
- What interaction the use case has with the actors
- What data is needed by the use case
- The description of any alternate or exceptional flows

The flow of events for a use case is contained in a document called the use case specification. Each project should use a standard template for the creation of the use case specification. Includes the following

1. Use case name – Brief Description
2. Flow of events –
    1. Basic flow
    2. Alternate flow
    3. Special requirements
    4. Pre conditions
    5. Post conditions
    6. Extension points

FLOW OF EVENTS FOR CHANGE PASSWORD

1.Use case: Change password for Admin.

Brief Description: This use case is started by Admin. It provides the capability for admin to change his/her password.

2.Actor: Admin

3.Flow events:

3.1 Basic flow:

***** This use case begins when admin logs into the system and enters his/her password. The system verifies that the password is valid.

- If the password is invalid alternate flow 3.3.1 is executed.

*Now, admin has to enter values of "current password", "new password", "confirm password", and clicks 'save' button.

-If the current password is wrong, alternate flow 3.3.2 is executed.

-If the current password and confirm password do not match alternate flow 3.3.3 is executed.

-After clicking save, database is updated and use case ends.

3.2 Alternate flow

3.3.1 An invalid password : Invalid password is entered by admin to login. Admin can re-enter a password or terminate the use case.

3.3.2 Wrong password : Wrong password is entered by admin while changing password. Admin can re-enter the password or terminate the use case.

3.3.3 Password mismatch : Values of "new password" and "confirm password" are mismatched. Admin can re-enter both or either of them which is wrong or terminate use case.

4.Pre condition : Admin should first login to the website.

5.Post condition : Next time, Admin can be able to login through new password.

FLOW OF EVENTS FOR VIEW ATTENDANCE BY USER

1.Use case : View attendance Report by user.

Brief Description : This use case is started by user. It provides the capability for user to view his/her attendance report either in detailed way or by dates wise.

2. Actor : User

3.Flow of events :

3.1 Basic flow :

***** This use case begins when user logs into the system and enters his/her password. The system verifies that the password is valid.

-If the password is invalid alternate flow 3.3.1 is executed.

Then, the user selects "view attendance report" .Now , the user is able to select "view detailed report" the report is generated and display on the system.

User can continue the use case or terminate use case.

If the user selects "view by dates" user needs to enter "start date" and "end date". System verifies whether 'start date' and 'end date' are valid or not.

If start date is not valid alternate flow 3.3.2 is executed.

If end date is not valid alternate flow 3.3.3 is executed.

System generates attendance report based on start date and end date and display it to user.

User can continue the use case or terminate the use case.

3.2 Alternate flow : 3.3.1 An invalid password : Invalid password is entered by user to login. User can re-enter a password or terminate the use case.

3.3.2 Invalid start date : Invalid start date is entered by user. User can re-enter the password or terminate the use case.

3.3.3 Invalid end date : Invalid end date is entered by user. User can re-enter the password or terminate the use case.

4.Pre condition : User should login to the portal.

5.Post condition : User can be able to view his/her attendance report.

7. BUILDING A BUSINESS PROCESS MODEL USING UML ACTIVITY DIAGRAM

An Activity diagram is a variation of a special case of a state machine, in which the states are activities representing the performance of operations and the transitions are triggered by the completion of the operations. The purpose of Activity diagram is to provide a view of flows and what is going on inside a use case or among several classes. You can also use activity diagrams to model code-specific information such as a class operation.

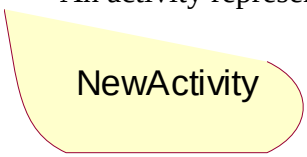
Activity diagrams are very similar to a flowchart because you can model a workflow from activity to activity. An activity diagram is basically a special case of a state machine in which most of the states are activities and most of the transitions are implicitly triggered by completion of the actions in the source activities.

- Activity Diagrams also may be created at this stage in the life cycle. These diagrams represent the dynamics of the system. They are flow charts that are used to show the workflow of a system; that is, they show the flow of control from activity to activity in the system, what activities can be done in parallel, and any alternate paths through the flow.
- At this point in the life cycle, activity diagrams may be created to represent the flow across use cases or they may be created to represent the flow within a particular use case.
- Later in the life cycle, activity diagrams may be created to show the workflow for an operation.

The following tools are used on the activity diagram toolbox to model activity diagrams:

Activities:

An activity represents the performance of some behavior in the workflow.



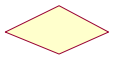
Transitions:

Transitions are used to show the passing of the flow of control from activity to activity. They are typically triggered by the completion of the behavior in the originating activity.

Decision Points:

When modeling the workflow of a system it is often necessary to show where the flow of control branches based on a decision point. The transitions from a decision point contain a guard condition, which is used to determine which path from the decision point

is taken. Decisions along with their guard conditions allow you to show alternate paths through a work flow.



Decision point

Start state:

A start state explicitly shows the beginning of a workflow on an activity diagram or the beginning of the execution of a state machine on a state chart diagram.

End state:

An End state represents a final or terminal state on an activity diagram or state chart diagram. Place an end state when you want to explicitly show the end of a workflow on an activity diagram or the end of a state chart diagram. Transitions can only occur into an end state; however, there can be any number of end states per context.

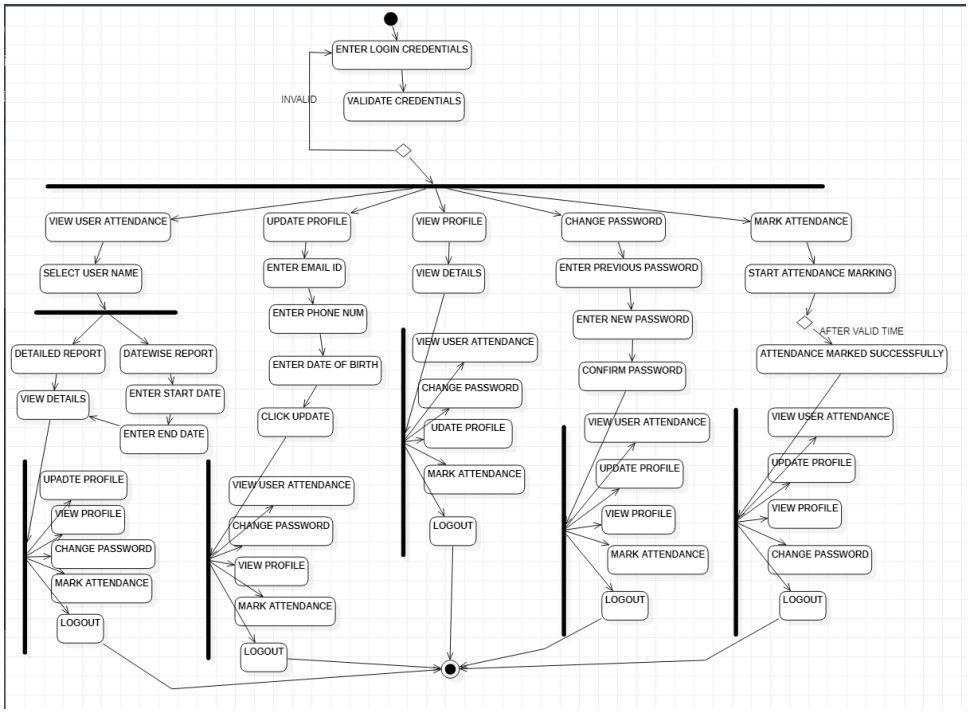
End state

Swim Lanes:

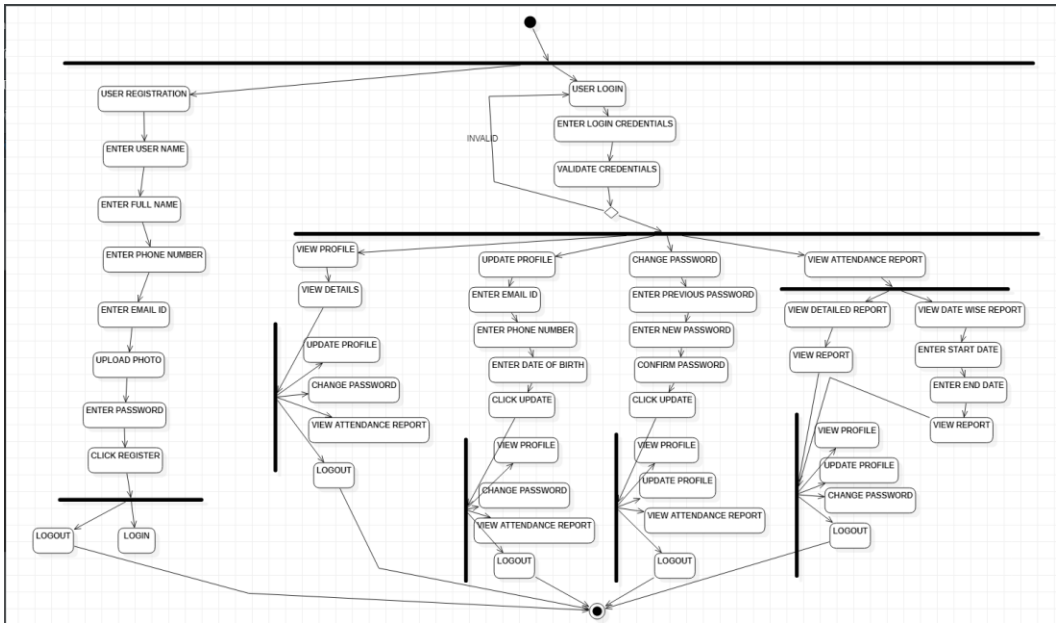
Swim lanes may be used to partition an activity diagram. This typically is done to show what person or organization is responsible for the activities contained in the swim lane.

- Horizontal synchronization
- Vertical synchronization.

Activity Diagram For ADMIN



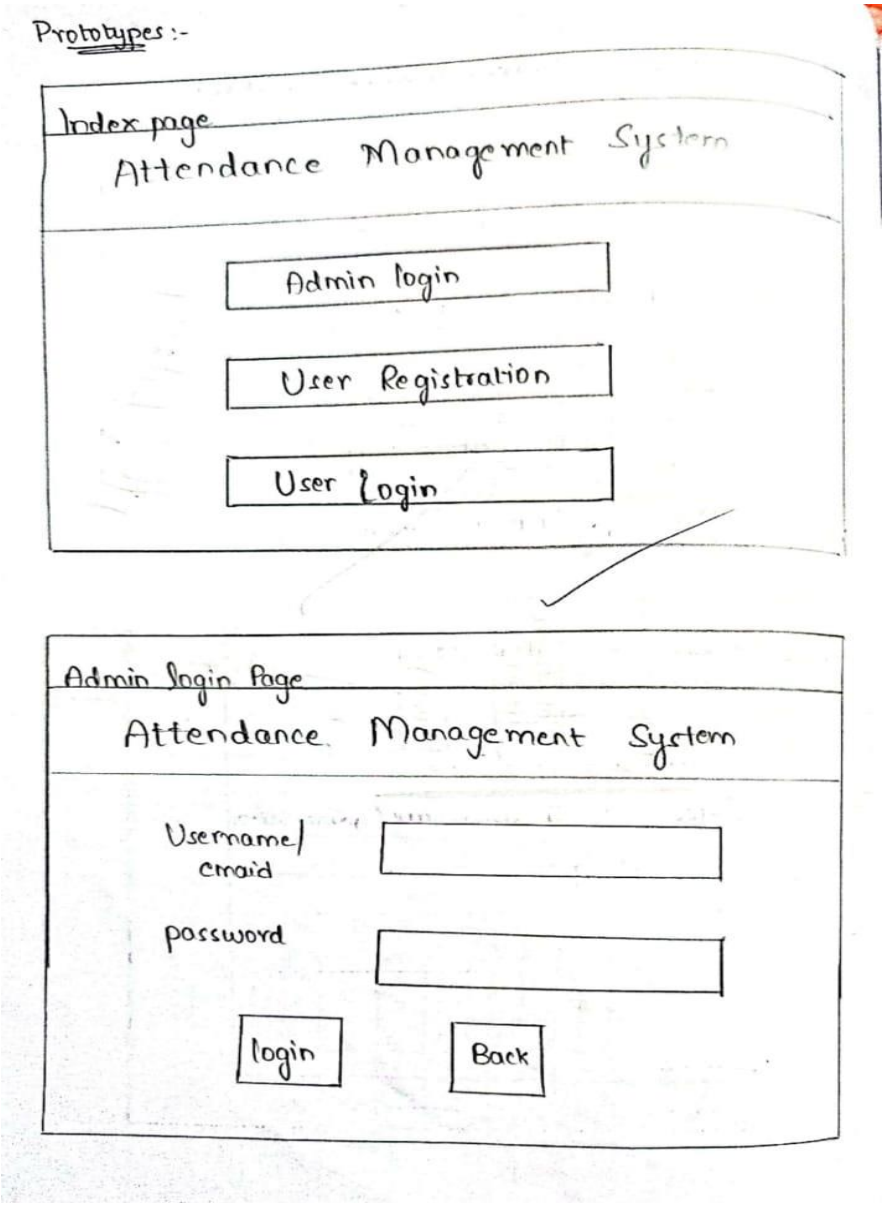
Activity diagram for USER



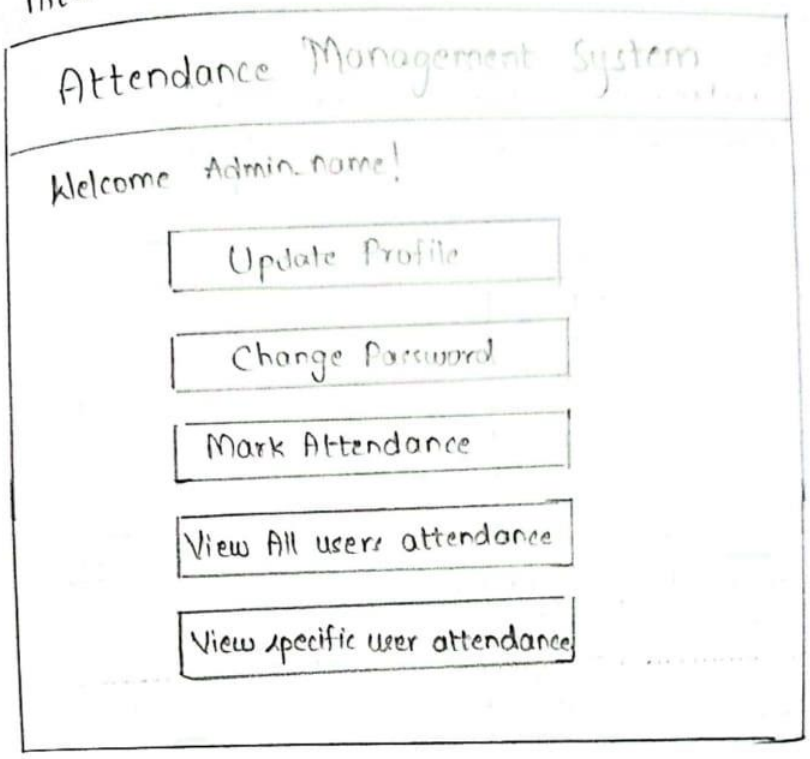
CONSTRUCTION OF PROTOTYPES

As the requirements for a system emerge in the form of use cases, it is sometimes helpful to build simple prototypes of how some of the use cases will work. A prototype is a working model of part of the system usually a program with limited functionality that is built to test out some aspect of how the system will work. Prototypes can be used to help elicit requirements. Showing users how the system might provide some of the use cases often produces a stronger reaction than showing them a series of abstract diagrams. Their reaction may contain useful information about requirements.

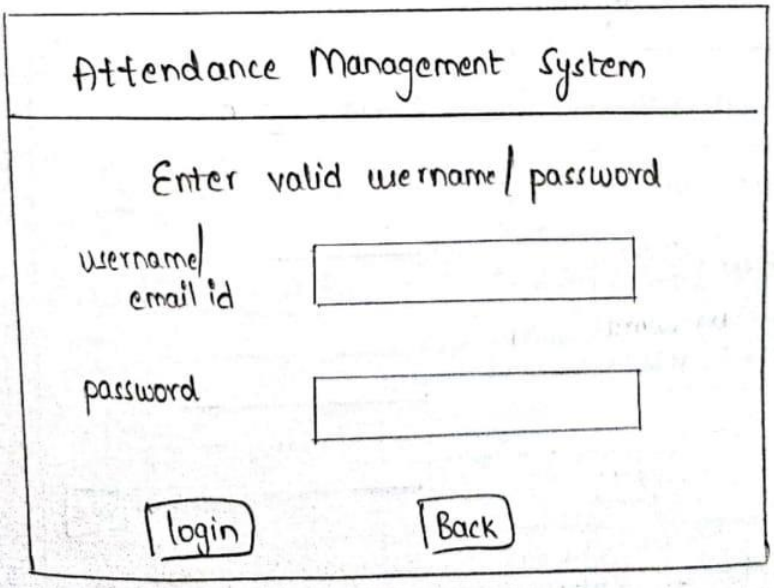
The following prototype is for FRAMS :



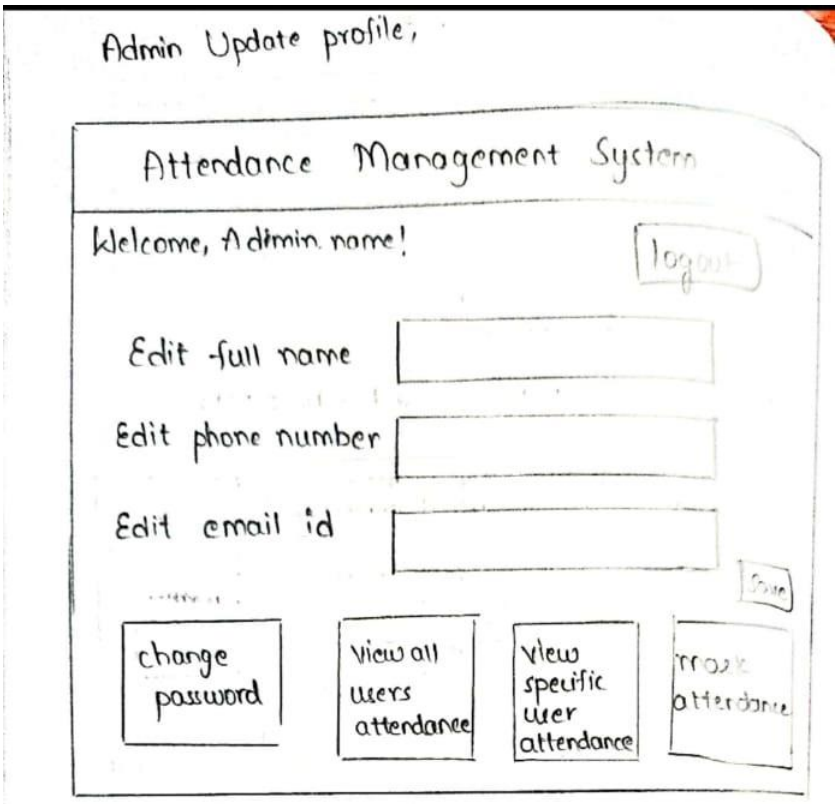
After admin Successful login



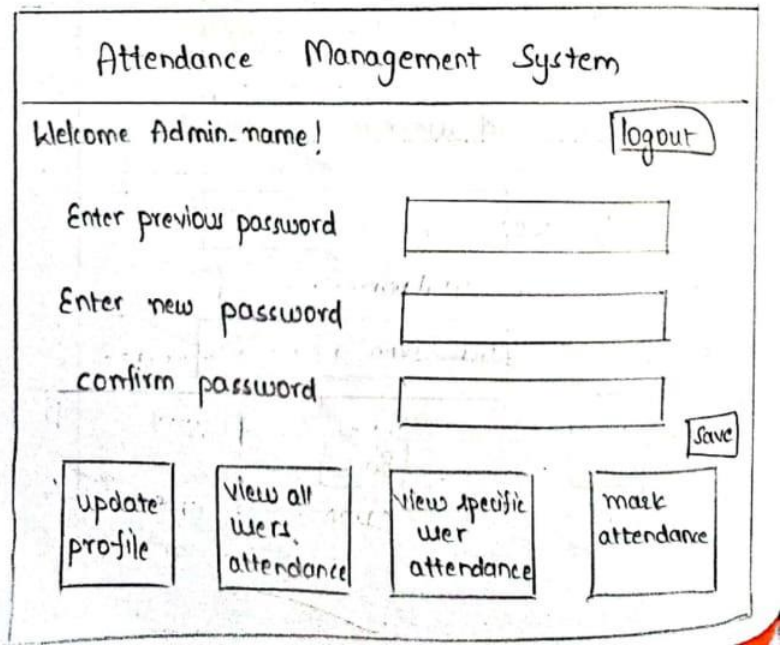
If Admin login was unsuccessful,



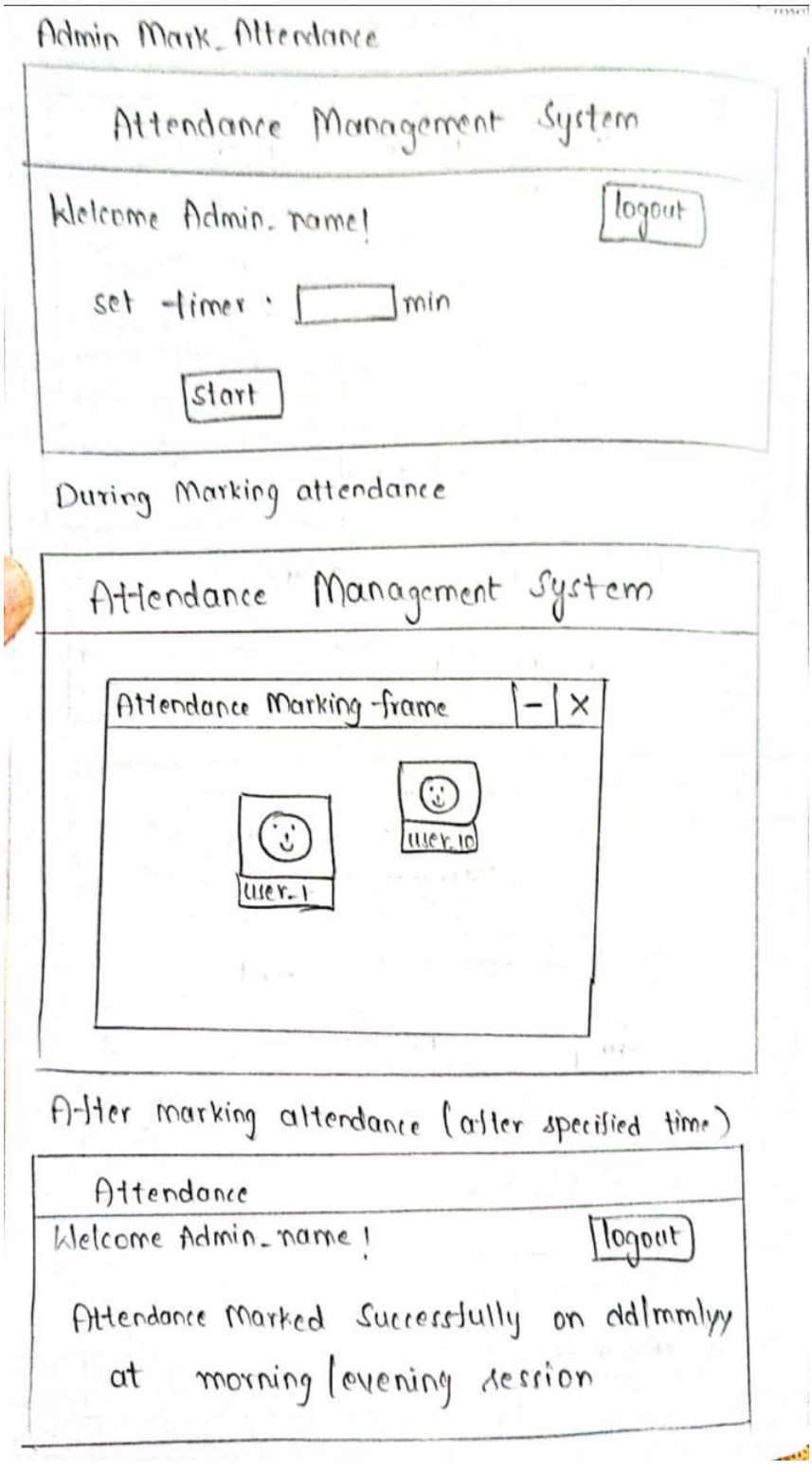
Admin Update profile,



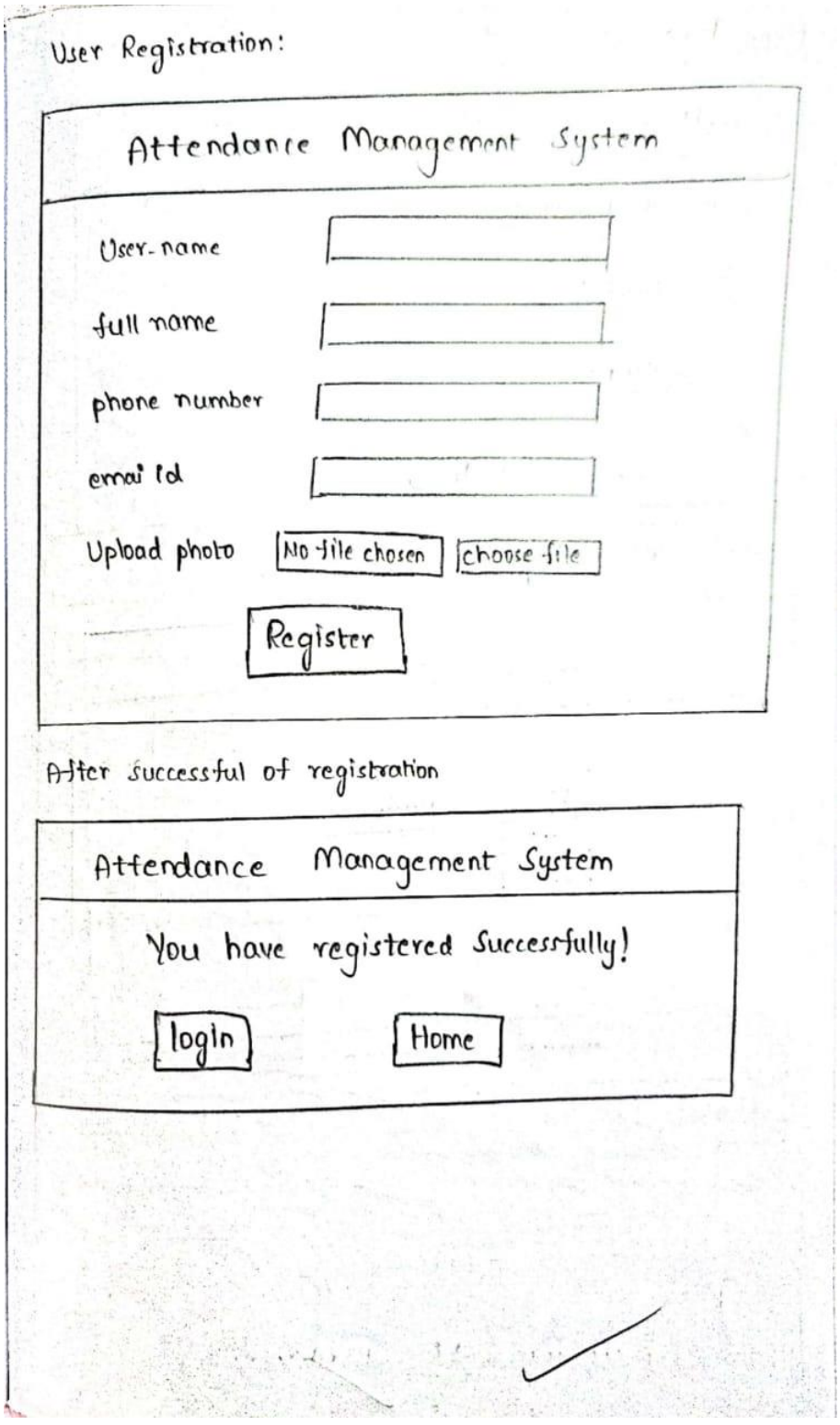
Admin change password,



Admin Mark_Attendance

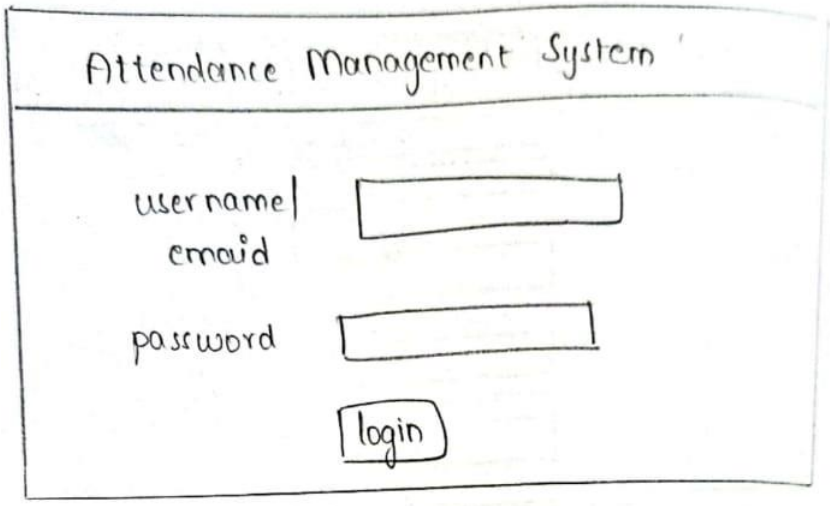


User Registration:

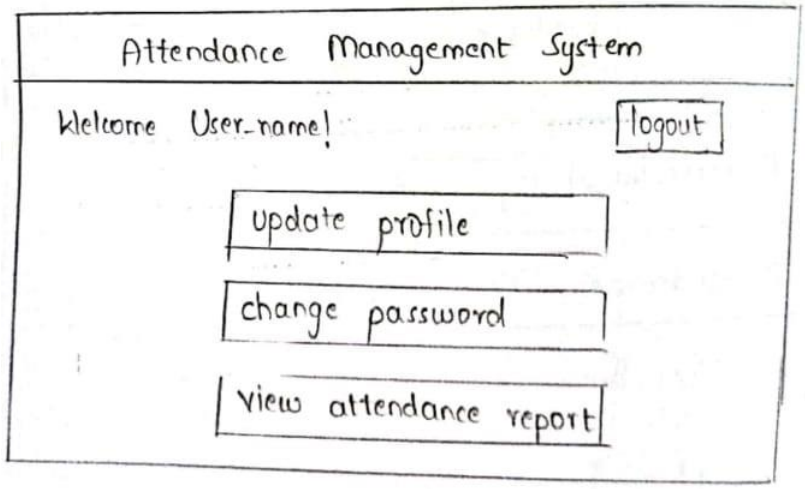


After successful of registration

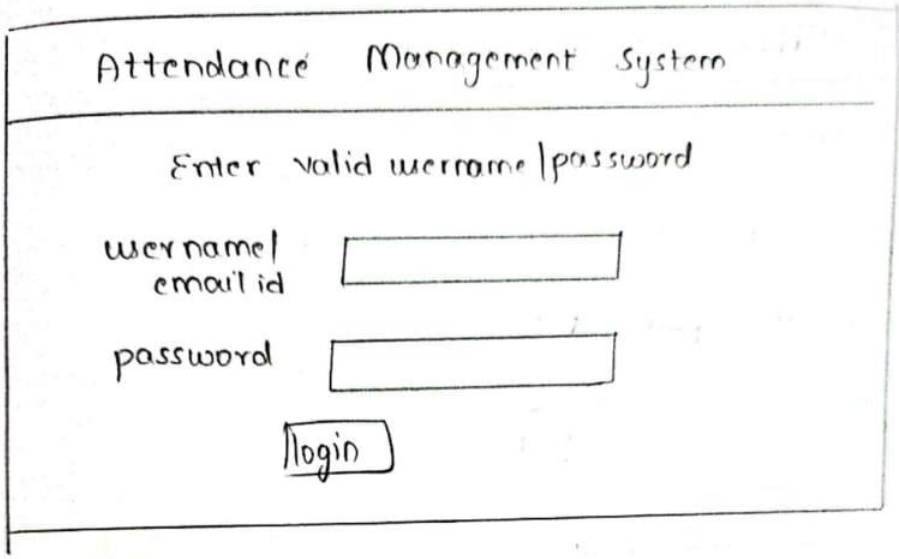
User login



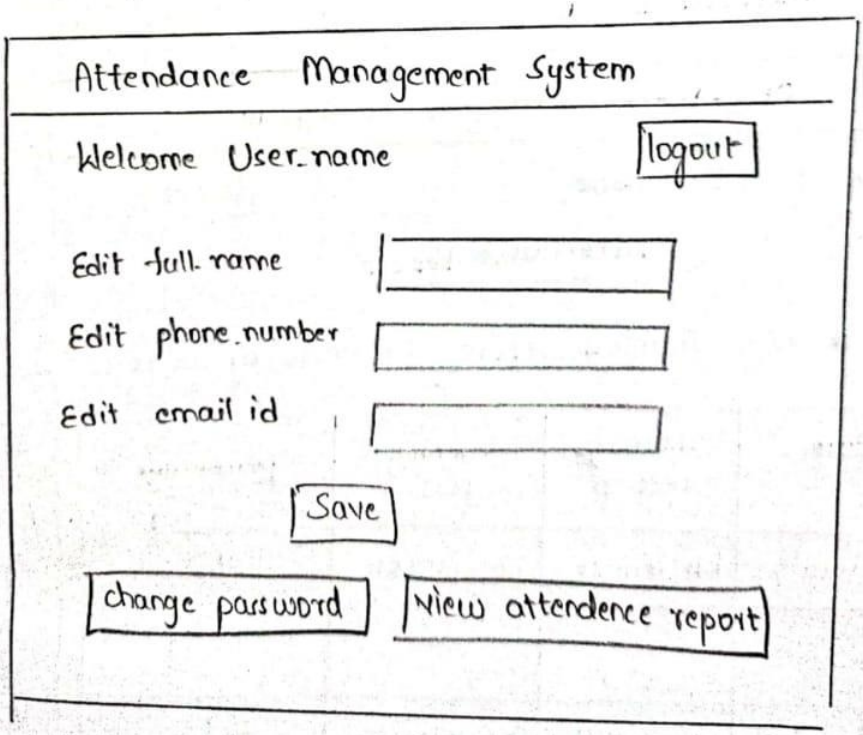
After successful login,



If login was unsuccesstul,



User update profile



User. change password

Attendance Management System

Logout

Previous password

New password

Confirm password

Save

update profile

view atterdance report

View attendance Report

Attendance Management System

Welcome User.name!

logout

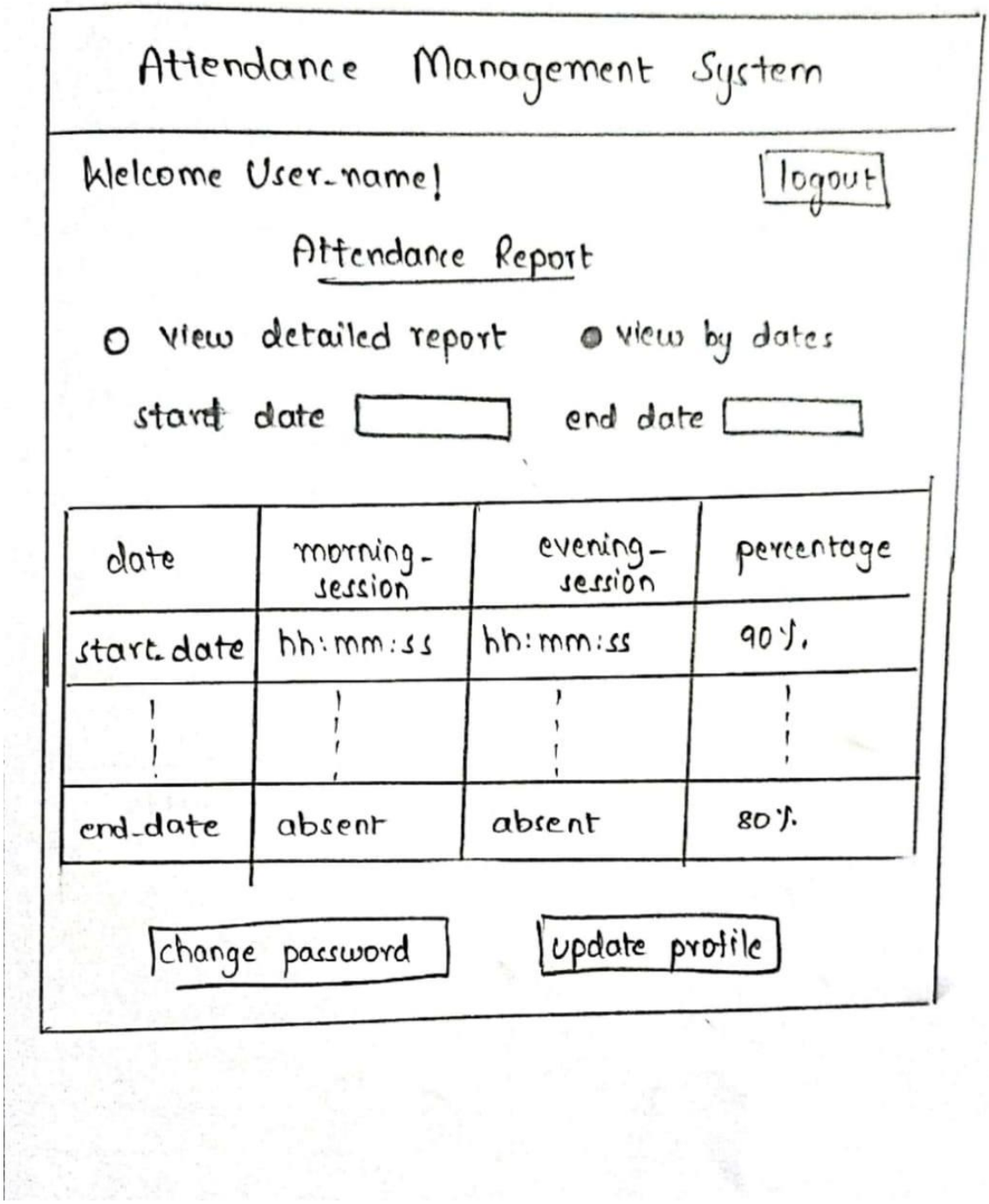
Attendance Report

● view detailed report ○ view by dates

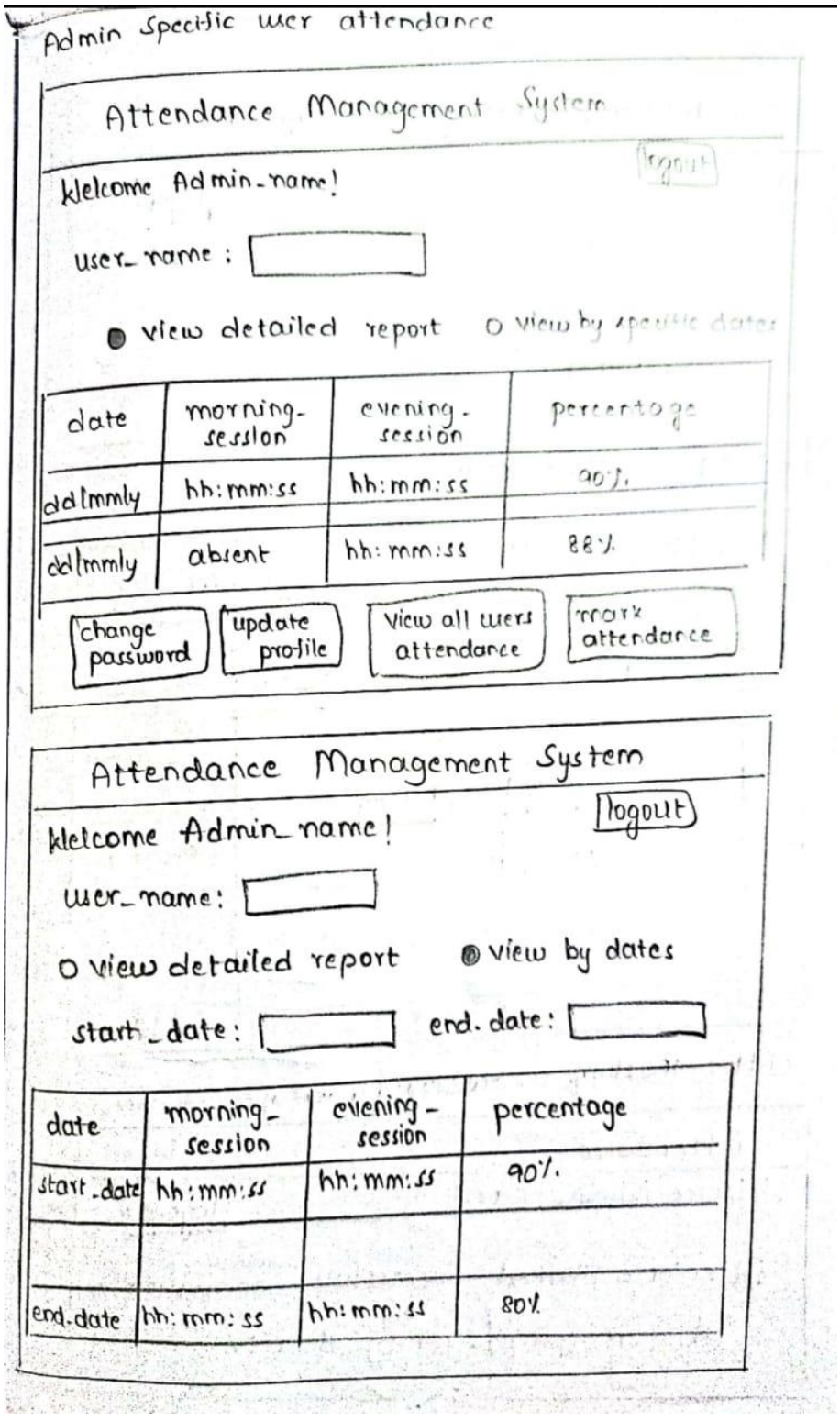
date	morning-session	evening-session	percentage
dd/mm/y	hh:mm:ss	hh:mm:ss	95%
dd/mm/yy	hh:mm:ss	hh:mm:ss	80%

change password

update profile


Attendance Management System
Welcome User_name!
logout
Attendance Report
view detailed report
view by dates
start date
end date
date
morning-session
evening-session
percentage
start_date
hh:mm:ss
hh:mm:ss
90%
end_date
absent
absent
80%
change password
update profile

Admin Specific user attendance



9. CONSTRUCTION OF SEQUENCE DIAGRAMS.

A sequence diagram is a graphical view of a scenario that shows object interaction in a time based sequence--what happens first, what happens next…

Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces.

A sequence diagram has two dimensions: the vertical dimension represents time; the horizontal dimension represents different objects. The vertical line is called the object's lifeline. The lifeline represents the object's existence during the interaction.

Steps:

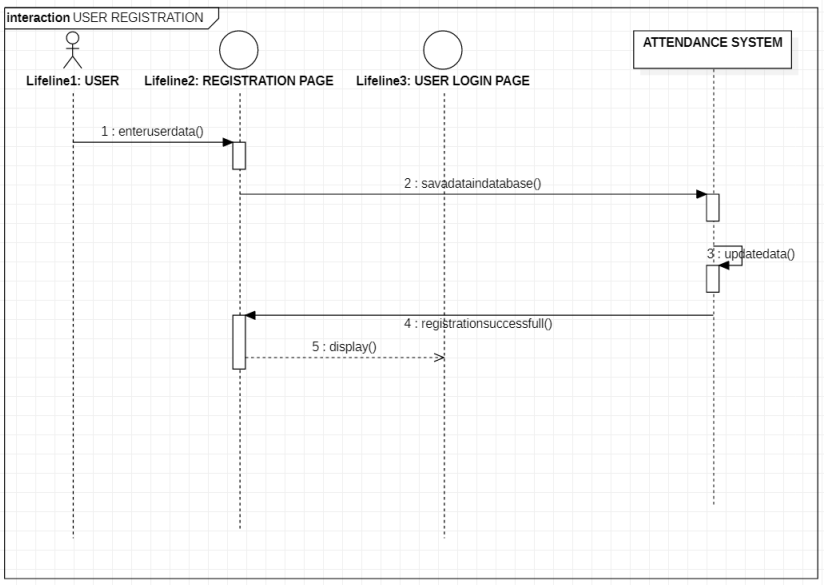
1. An object is shown as a box at the top of a dashed vertical line. Object names can be specific (e.g., Algebra 101, Section 1) or they can be general (e.g., a course offering). Often, an anonymous object (class name may be used to represent any object in the class.)
2. Each message is represented by an Arrow between the lifelines of two objects. The order in which these messages occur is shown top to bottom on the page. Each message is labeled with the message name.

There are two main differences between sequence and collaboration diagrams: sequence diagrams show time-based object interaction while collaboration diagrams show how objects associate with each other. A sequence diagram has two dimensions: typically, vertical placement represents time and horizontal placement represents different objects.

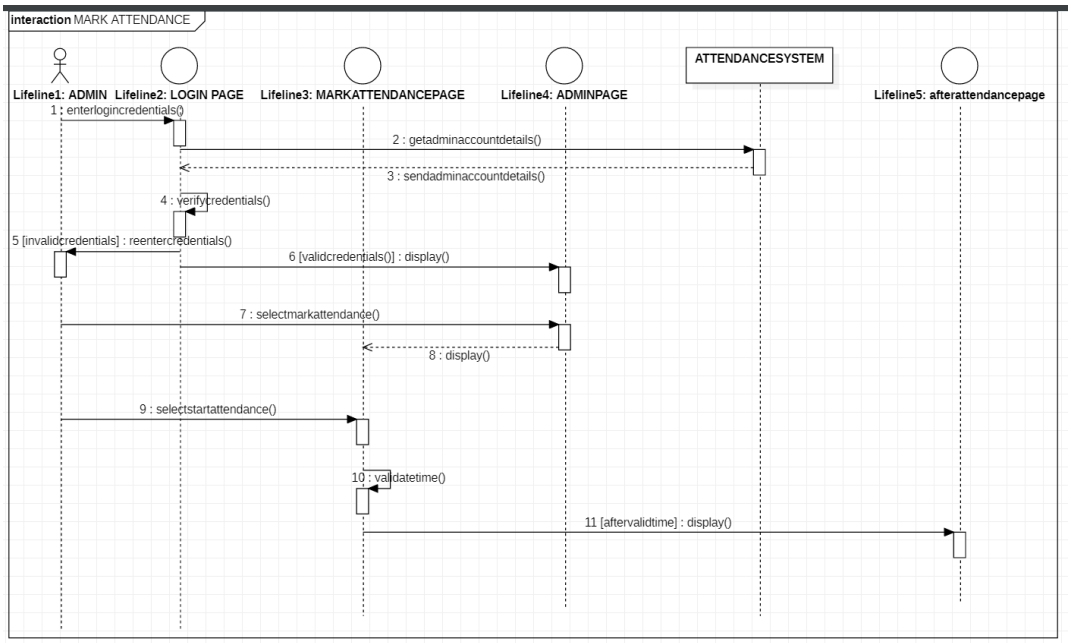
ELEMENTS OF SEQUENCE DIAGRAM:

- Objects
- Links
- Messages
- Focus of control
- Object life line

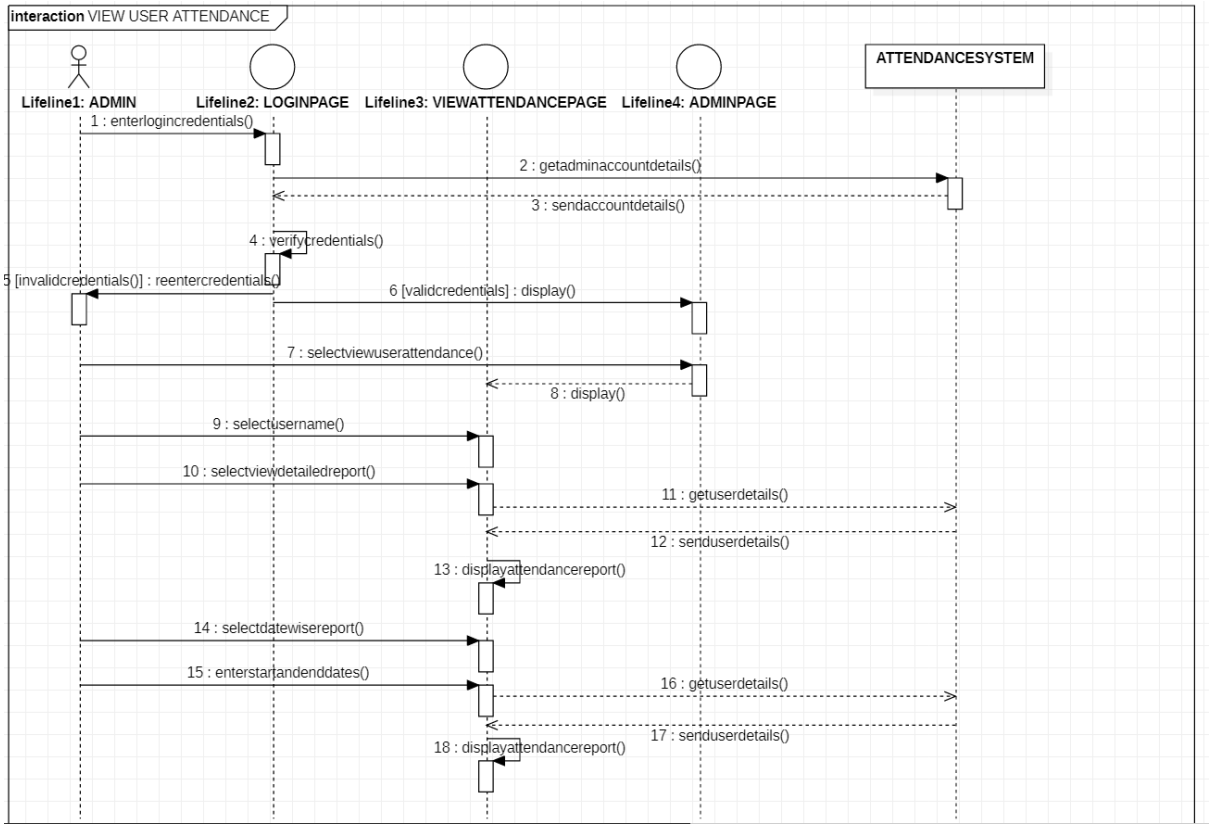
SEQUENCE DIAGRAM FOR USER REGISTRATION



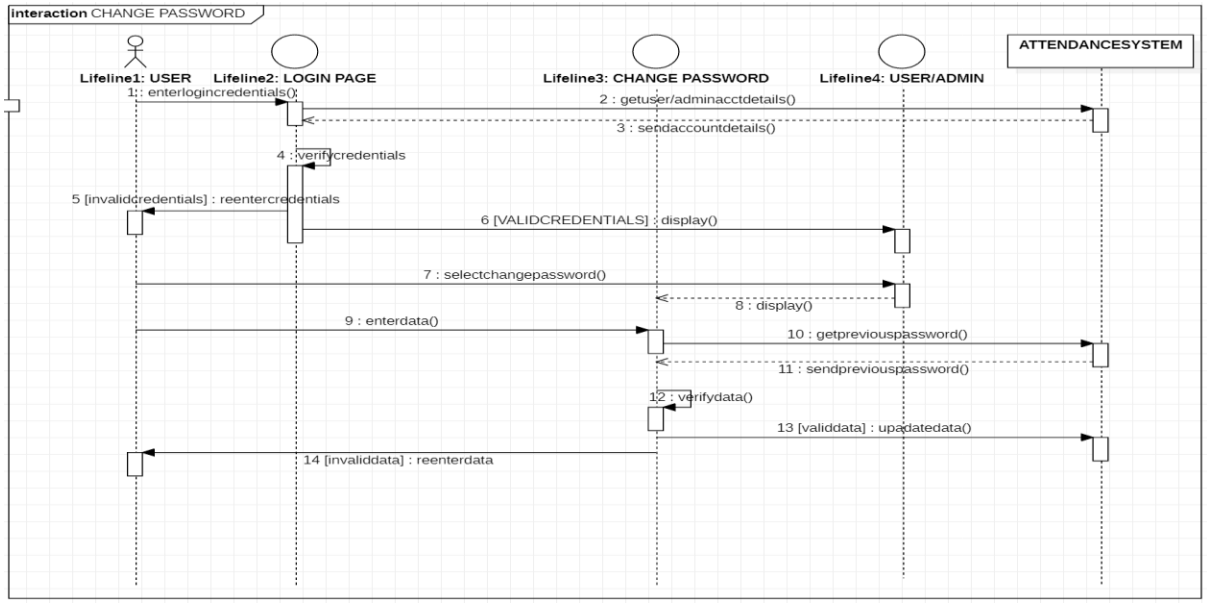
SEQUENCE DIAGRAM FOR MARK ATTENDANCE



SEQUENCE DIAGRAM FOR VIEW USER ATTENDANCE



SEQUENCE DIAGRAM FOR CHANGE PASSWORD



10. CONSTRUCTION OF COLLABORATION DIAGRAMS

Collaboration diagrams are the second kind of interaction diagram in the UML diagrams. They are used to represent the collaboration that realizes a use case. The most significant difference between the two types of interaction diagram is that a collaboration diagram explicitly shows the links between the objects that participate in a collaboration , as in sequence diagrams, there is no explicit time dimension.

Message labels in collaboration diagrams:

Messages on a collaboration diagram are represented by a set of symbols that are the same as those used in a sequence diagram, but with some additional elements to show sequencing and recurrence as these cannot be inferred from the structure of the diagram. Each message label includes the message signature and also a sequence number that reflects call nesting, iteration, branching, concurrency and synchronization within the interaction.

The formal message label syntax is as follows:

[predecessor] [guard-condition] sequence-expression [return-value ':='] message-name' (' [argument-list] ')'

A *predecessor* is a list of sequence numbers of the messages that must occur before the current message can be enabled. This permits the detailed specification of branching pathways. The message with the immediately preceding sequence number is assumed to be the predecessor by default, so if an interaction has no alternative pathways the predecessor list may be omitted without any ambiguity. The syntax for a predecessor is as follows:

sequence-number { ',' sequence-number} *'I'*

The *'I'* at the end of this expression indicates the end of the list and is only included when an explicit predecessor is shown.

Guard conditionsare written in Object Constraint Language (OCL) ,and are only shown where the enabling of a message is subject to the defined condition. A guard condition may be used to represent the synchronization of different threads of control.

A *sequence-expression*is a list of integers separated by dots ('.') optionally followed by a *name* (a single letter), optionally followed by a *recurrence* term and terminated by a colon. A sequence-expression has the following syntax:

integer { '.' integer } [name] [recurrence] ':'

In this expression *integer* represents the sequential order of the message. This may be nested within a loop or a branch construct, so that, for example, message 5.1 occurs after message 5.2 and both are contained within the activation of message 5.

The *name* of a sequence-expression is used to differentiate two concurrent messages since these are given the same sequence number. For example, messages 3.2.1a and 3.2.1b are concurrent within the activation of message 3.2.

Recurrence reflects either iterative or conditional execution and its syntax is as follows:

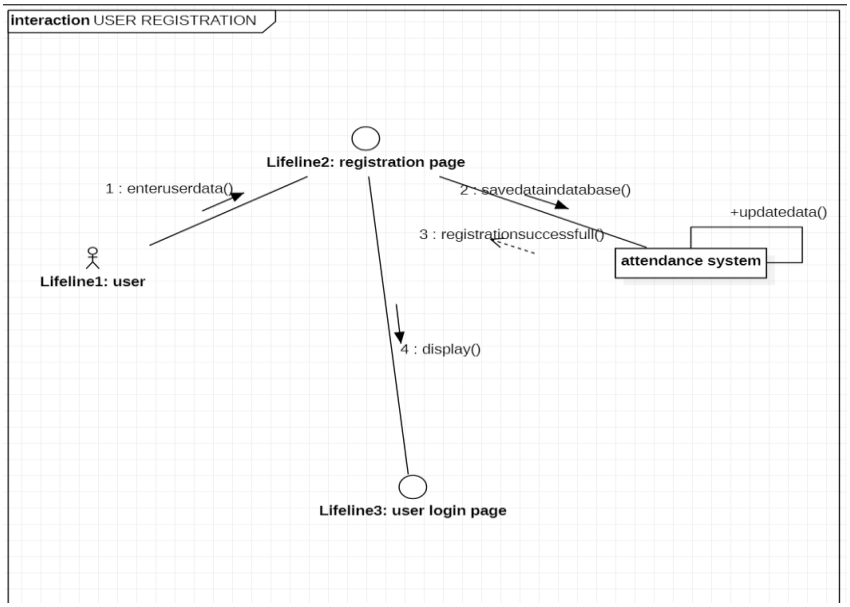
Branching: '['condition-clause‘] ,

Iteration: ‘ * ‘ ‘ [‘ iteration-clause ‘] '

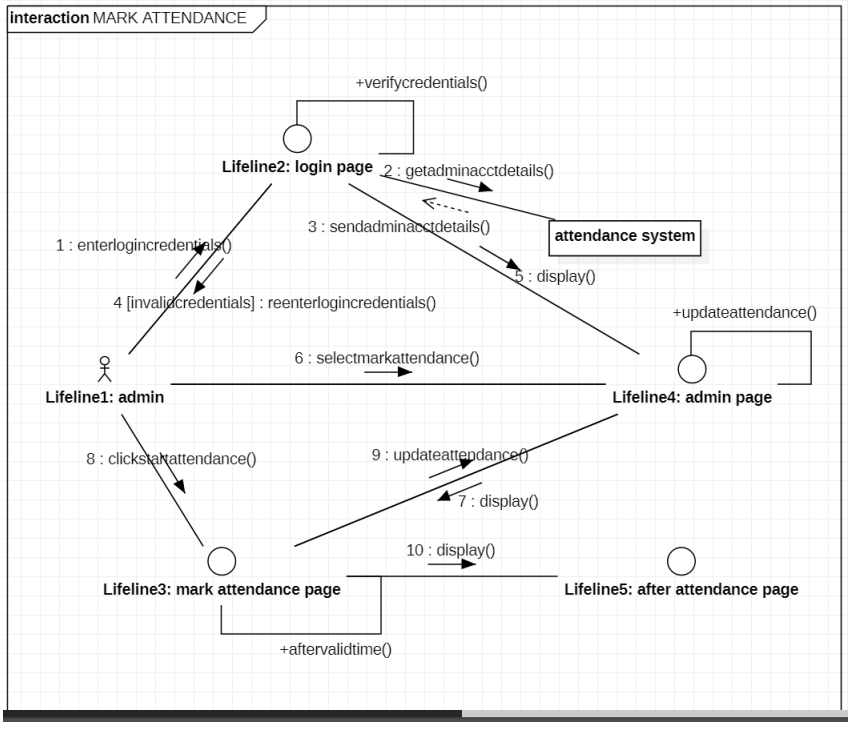
Elements:

- Objects ,Messages ,Path,Sequence Numbers,Links

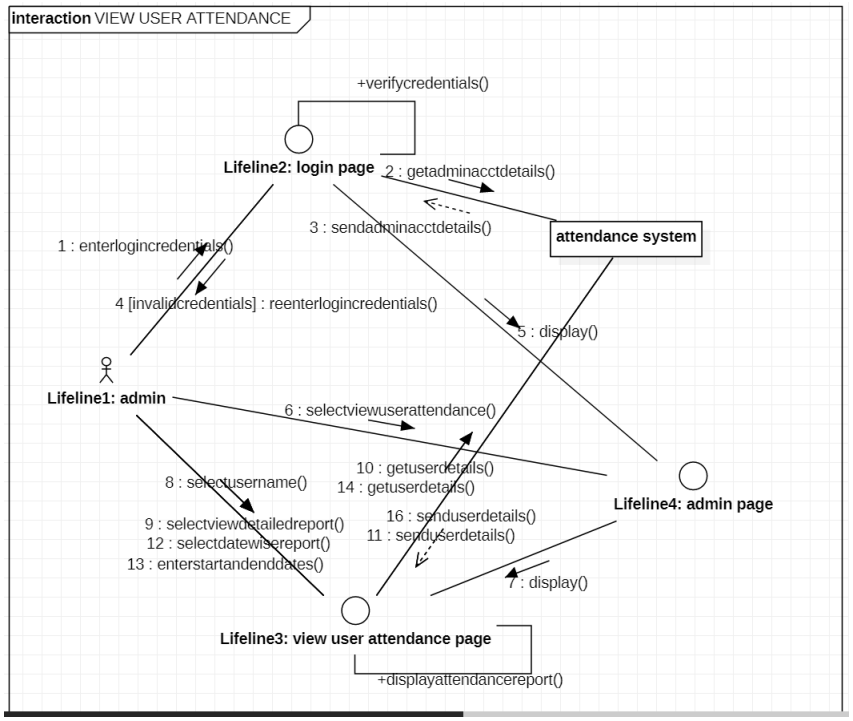
COLLABORATION DIAGRAM FOR USER REGISTRATION



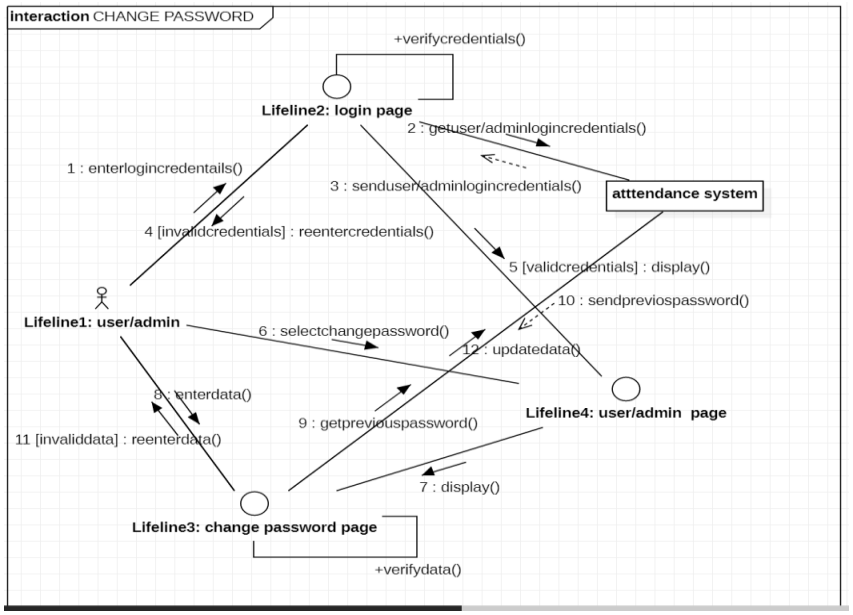
COLLABORATION DIAGRAM FOR MARK ATTENDANCE



COLLABORATION DIAGRAM FOR VIEW USER ATTENDANCE



COLLABORATION DIAGRAM FOR CHANGE PASSWORD



11. Construction of UML static class diagram

Class diagrams contain icons representing classes, packages, interfaces, and their relationships. You can create one or more class diagrams to depict the classes at the top level of the current model; such class diagrams are themselves contained by the top level of the current model.

Class:

A Class a description of a group of objects with common properties (attributes), common behavior (operations), common relationships to other objects, and common semantics.

Thus, a class is a template to create objects. Each object is an instance of some class and objects cannot be instances of more than one class.

Classes should be named using the vocabulary of the domain.

For example, the CourseOffering class may be defined with the following characteristics:

Attributes - location, time offered

Operations - retrieve location, retrieve time of day, add a student to the offering .

Each object would have a value for the attributes and access to the operations specified by the CourseOffering class.

In the UML, classes are represented as compartmentalized rectangles.

The top compartment contains the name of the class.

The middle compartment contains the structure of the class (attributes).

The bottom compartment contains the behavior of the class (operations) as shown below.

OBJECT :

- AN OBJECT IS a representation of an entity, either real-world or conceptual.
- An object is a concept, abstraction, or thing with well defined boundaries and meaning for an application.
- Each object in a system has three characteristics: state, behavior, and identity.

STATE : THE STATE OF an object is one of the possible conditions in which it may exist. The state of an object typically changes over time, and is defined by a set of properties (called attributes), with the values of the properties, plus the relationships the object may have with other objects.

For example, a course offering object in the registration system may be in one of two states: *open* and *closed.* It is available in the open state if value is < 10 otherwise closed.

Behavior :

- Behavior determines how an object responds to requests from other objects .
- Behavior is implemented by the set of operations for the object.

For example , In the registration system, a course offering could have the behaviors *add* a *student* and *delete* a *student.*

Identity :

- Identity means that each object is unique even if its state is identical to that of another object.

Attributes

Attributes are part of the essential description of a class. They belong to the class, unlike objects, which instantiate the class. Attributes are the common structure of what a member of the class can 'know'. Each object will have its own, possibly unique, value for each attribute.

Guidelines for identifying attributes of classes are as follows:

- Attributes usually correspond to nouns followed by prepositional phrases
- Keep the class simple; state only enough attribute to defineobject state.
- Attributes are less likely to be fully described in the problem statement.
- Omit derived attributes.
- Do not carry discovery attributes to excess.

STEREOTYPES AND CLASSES :

As like stereotypes for relationships in use case diagrams. Classes can also have stereotypes. Here a stereotype provides the capability to create a new kind of modeling element. Here, we can create new kinds of classes. Some common stereotypes for a class are entity Class, boundary Class, control class, and exception.

Entity Classes

- An **entity class** models information and associated behavior that is generally long lived.
- This type of class may reflect a real-world entity or it may be needed to perform tasks internal to the system.
- They are typically independent of their surroundings; that is, they are not sensitive to how the surroundings communicate with the system.

Boundary Classes :

Boundary classes handle the communication between the system surroundings and the inside of the system. They can provide the interface to a user or another system (i.e., the interface to an actor). They constitute the surroundings dependent part of the system. Boundary classes are used to model the system interfaces.

Boundary classes are also added to facilitate communication with other systems. During design phase, these classes are refined to take into consideration the chosen communication protocols.

Control Classes

- Control classes model sequencing behavior specific to one or more use cases.

- Control classes coordinate the events needed to realize the behavior specified in the use case.
- Control classes typically are application-dependent classes.

  In the early stages of the Elaboration Phase, a control class is added for each actor/use case pair. The control class is responsible for the flow of events in the use case.

  In the early stages of the Elaboration Phase, a control class is added for each actor/use case pair. The control class is responsible for the flow of events in the use case.

NEED FOR RELATIONSHIPS AMONG CLASSES:

All systems are made up of many classes and objects. System behaviour is achieved through the collaborations of the objects in the system.

Two types of relationships in CLASS diagram are:

1. Association Relationship
2. Aggregation Relationship

1. Association Relationship:

An association is a bidirectional semantic connection between classes. It is not a data flow as defined in structured analysis and design data may flow in either direction across the association. An association between classes means that there is a link between objects in the associated classes.

2. Aggregation Relationship:

An aggregation relationship is a specialized form of association in which a whole is related to its part(s). Aggregation is known as a "part-of" or containment relationship. The UML notation for an aggregation relationship is an association with a diamond next to the class denoting the aggregate(whole).

3. Super-sub structure (Generalization Hierarchy):

These allow objects to be build from other objects. The super-sub class hierarchy is a relationship between classes, where one class is the parent class of another class.

NAMING RELATIONSHIP:

An association may be named. Usually the name is an active verb or verb phrase that communicates the meaning of the relationship. Since the verb phrase typically implies a reading direction, it is desirable to name the association so it reads correctly from left to right or top to bottom. The words may have to be changed to read the association in the other direction (e.g., Buses are allotted to Routes). It is important to note that the name of the association is optional.

ROLE NAMES:

The end of an association where it connects to a class is called an association role. Role names can be used instead of association names.

A role name is a noun that denotes how one class associates with another. The role name is placed on the association near the class that it modifies, and may be placed on one or both ends of an association line.

- It is not necessary to have both a role name and an association name.

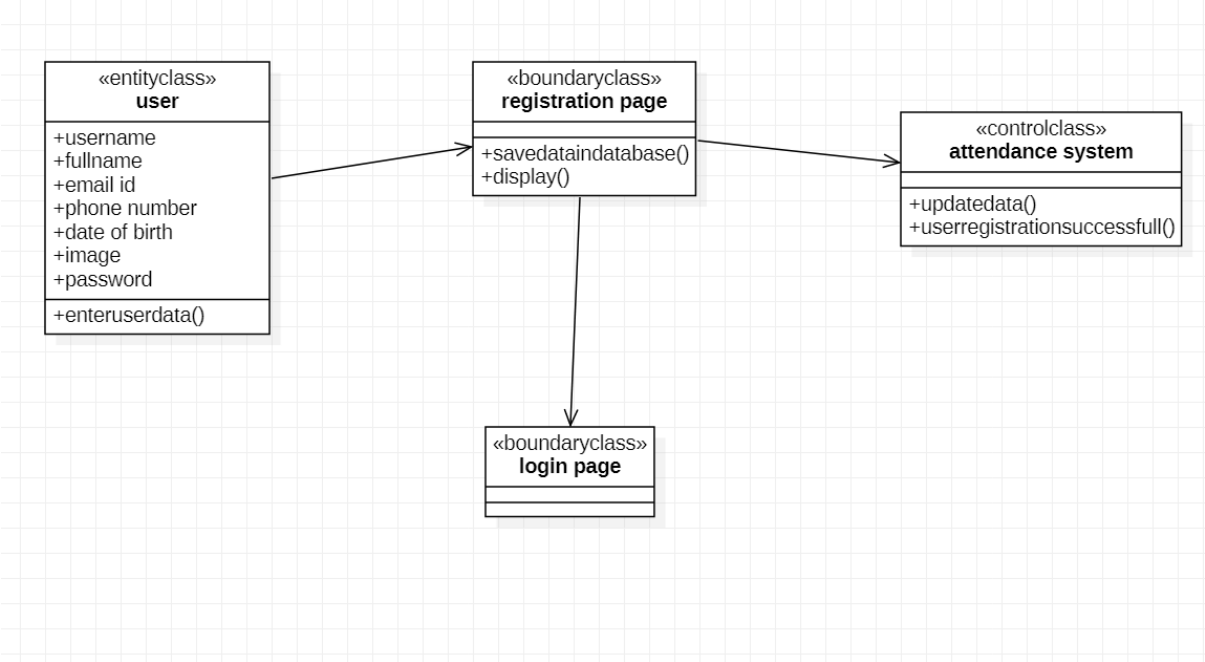
- Associations are named or role names are used only when the names are needed for clarity.

MULTIPLICITY INDICATORS:

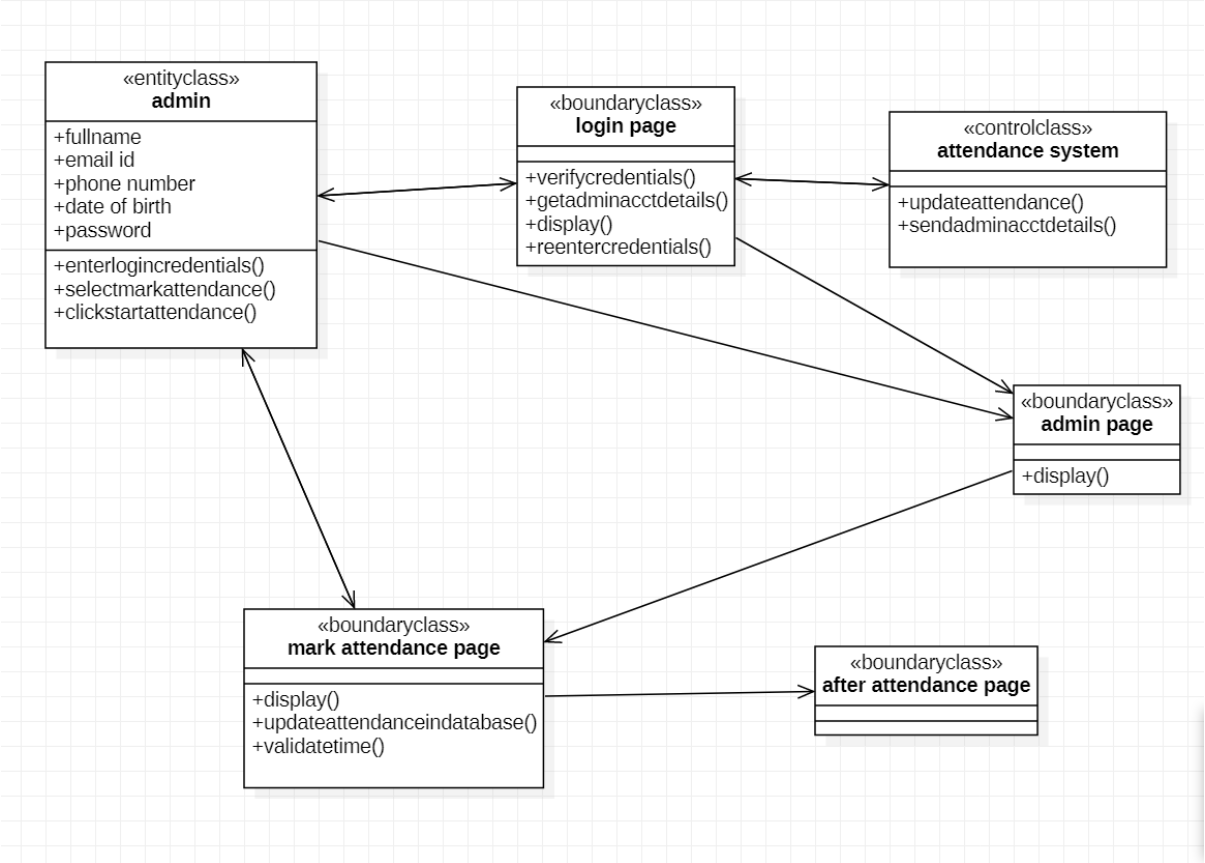
Although multiplicity is specified for classes, it defines the number of objects that participate in a relationship. Multiplicity defines the number of objects that are linked to one another. There are two multiplicity indicators for each association or aggregation one at each end of the line. Some common multiplicity indicators are

1	Exactly one
0... *	Zero or more
1... *	One or more
0... 1	Zero or one
5... 8	Specific range (5, 6, 7, or 8)
4... 7, 9	Combination (4, 5, 6, 7, or 9)

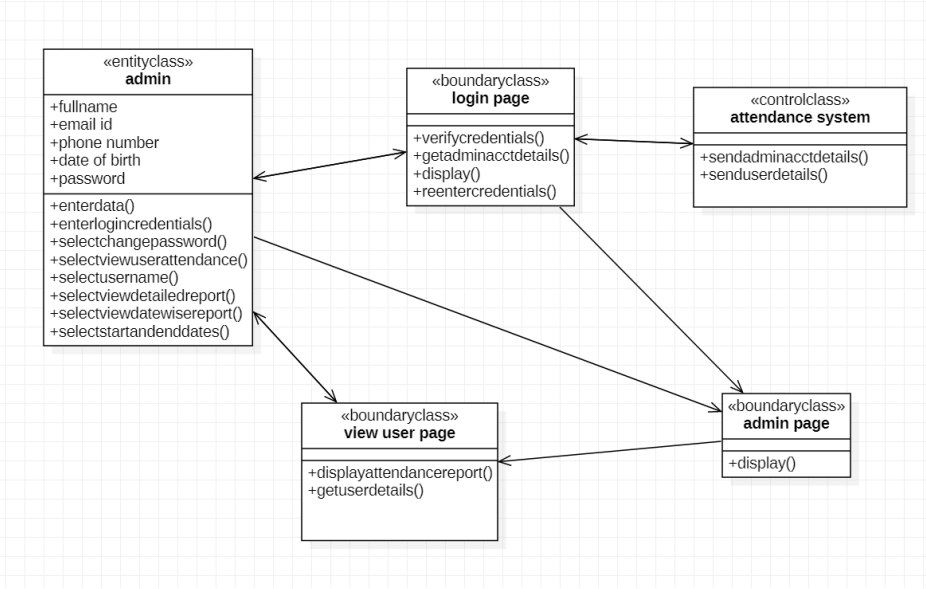
CLASS DIAGRAM FOR USER REGISTRATION



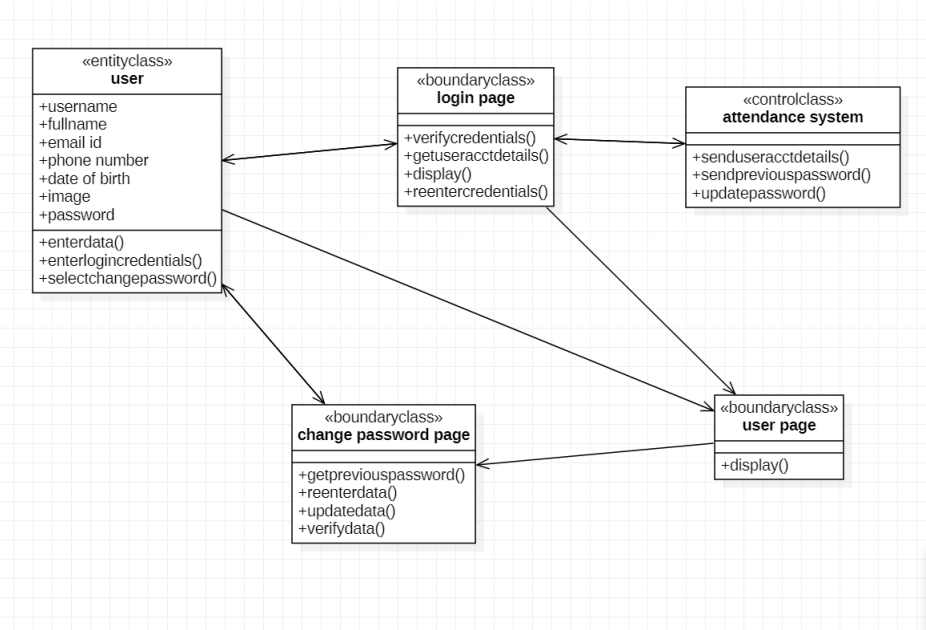
CLASS DIAGRAM FOR MARK ATTENDANCE



CLASS DIAGRAM FOR VIEW USER ATTENDANCE



CLASS DIAGRAM FOR CHANGE PASSWORD



12. Analyzing the object behavior by constructing the UML State Chart diagram

Use cases and scenarios provide a way to describe system behavior; in the form of interaction between objects in the system. Sometimes it is necessary to consider inside behavior of an object.

A state chart diagram shows the **states** of a single object, the events or messages that cause a **transition** from one state to another, and the **actions** that result from a state change. As in Activity diagram , state chart diagram also contains special symbols for start state and stop state.

State chart diagram cannot be created for every class in the system , it is only for those class objects with significant behavior.

State chart diagrams are closely related to activity diagrams. The main difference between the two diagrams is state chart diagrams are state centric, while activity diagrams are activity centric. A state chart diagram is typically used to model the discrete stages of an object's lifetime, whereas an activity diagram is better suited to model the sequence of activities in a process.

STATE:

A state represents a condition or situation during the life of an object during which it satisfies some condition, performs some action or waits for some event.

UML notation for STATE is

To identify the states for an object its better to concentrate on sequence diagram. In an ESU the object for Course Offering may have in the following states, initialization, open and closed state. These states are obtained from the attribute and links defined for the object. Each state also contains a compartment for actions.

Actions:

Actions on states can occur at one of four times:

- on entry
- on exit
- do
- on event.

on entry : What type of action that object has to perform after entering into the state.

on exit : What type of action that object has to perform after exiting from the state.

Do : The task to be performed when object is in this state, and must to continue until it leaves the state.

on event : An on event action is similar to a state transition label with the following syntax: event(args)[condition] : the Action

State Transition:

A state transition indicates that an object in the source state will perform certain specified actions and enter the destination state when a specified event occurs or when certain conditions are satisfied. A state transition is a relationship between two states, two activities, or between an activity and a state. You can show one or more state

transitions from a state as long as each transition is unique. Transitions originating from a state cannot have the same event, unless there are conditions on the event.

Transitions are labeled with the following syntax:

event (arguments) [condition] / action ^ target. send Event (arguments)

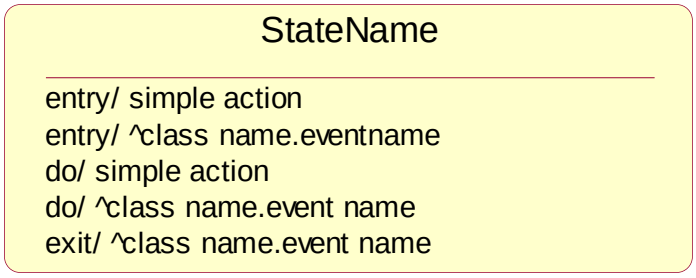
Only one event is allowed per transition, and one action per event.

State Details :

Actions that accompany all state transitions into a state may be placed as an entry action within the state. Like wise that accompany all state transitions out of a state may be placed as exit actions within the state. Behavior that occurs within the state is called an activity.

An activity starts when the state is entered and either completes or is interrupted by an outgoing state transition. The behavior may be a simple action or it may be an event sent to another object.

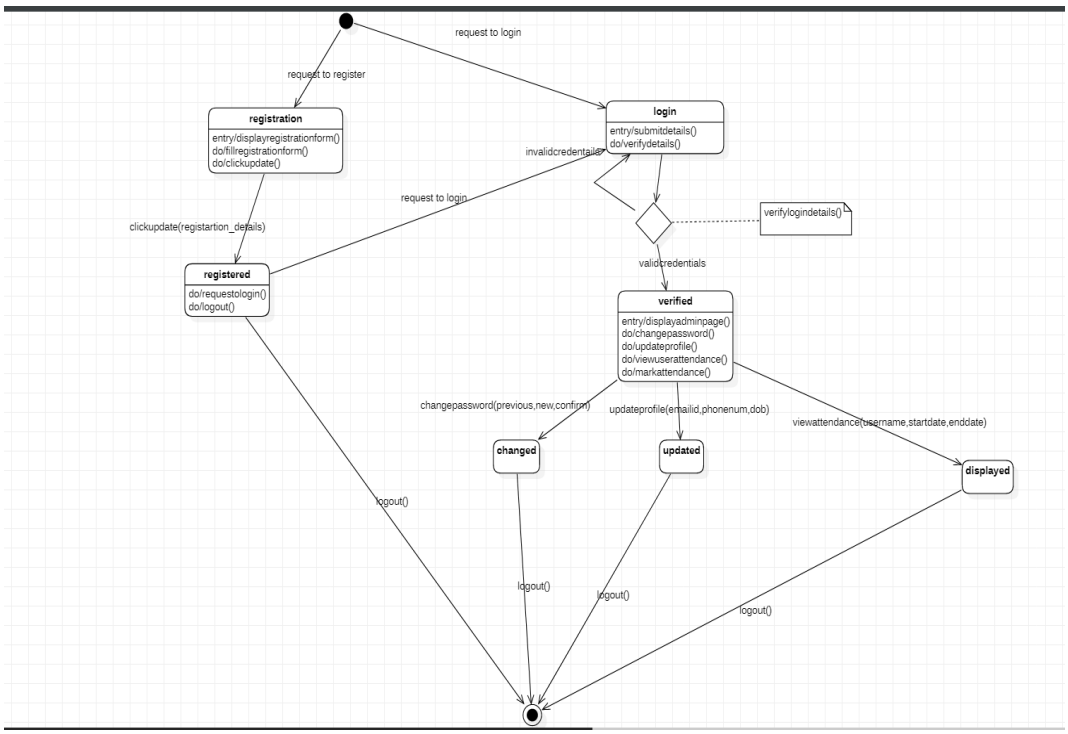
UML notation for State Details:



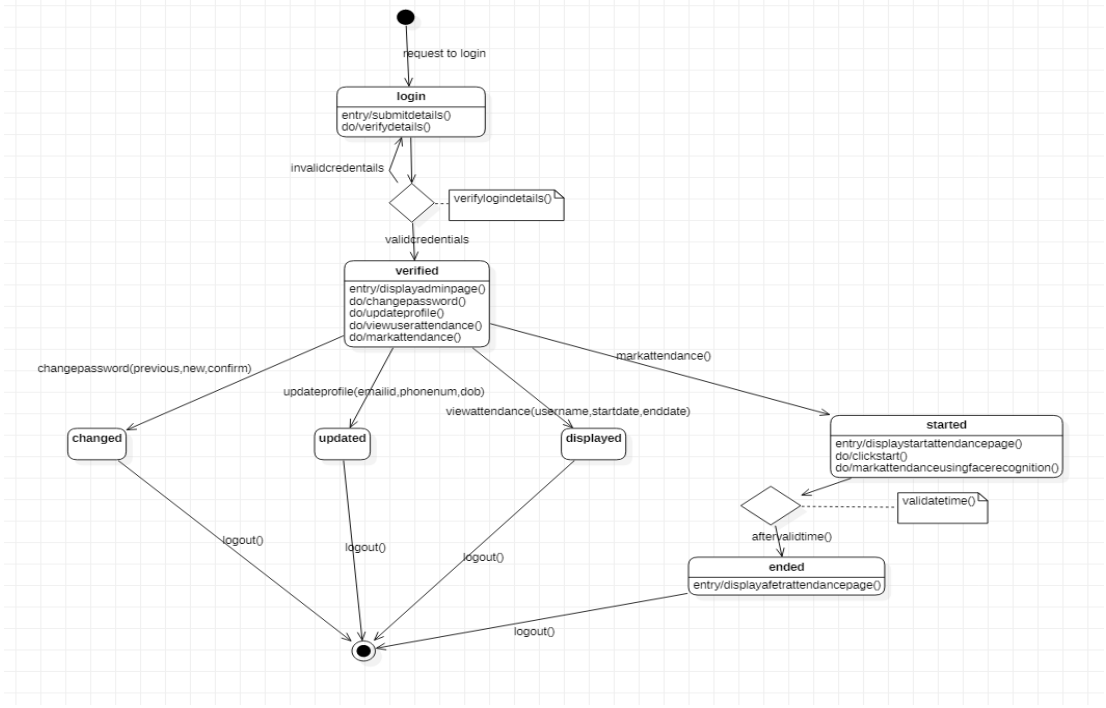
Purpose of State chart diagram:

- State chart diagrams are used to model dynamic view of a system.
- State chart diagrams are used to modelling lifetime of an object.
- State chart diagrams are used to focus on the changing state of a system driven by events.
- It will also be used when showing the behavior of a class over several use cases.

STATE CHART DIAGRAM FOR USER



STATE CHART DIAGRAM FOR ADMIN



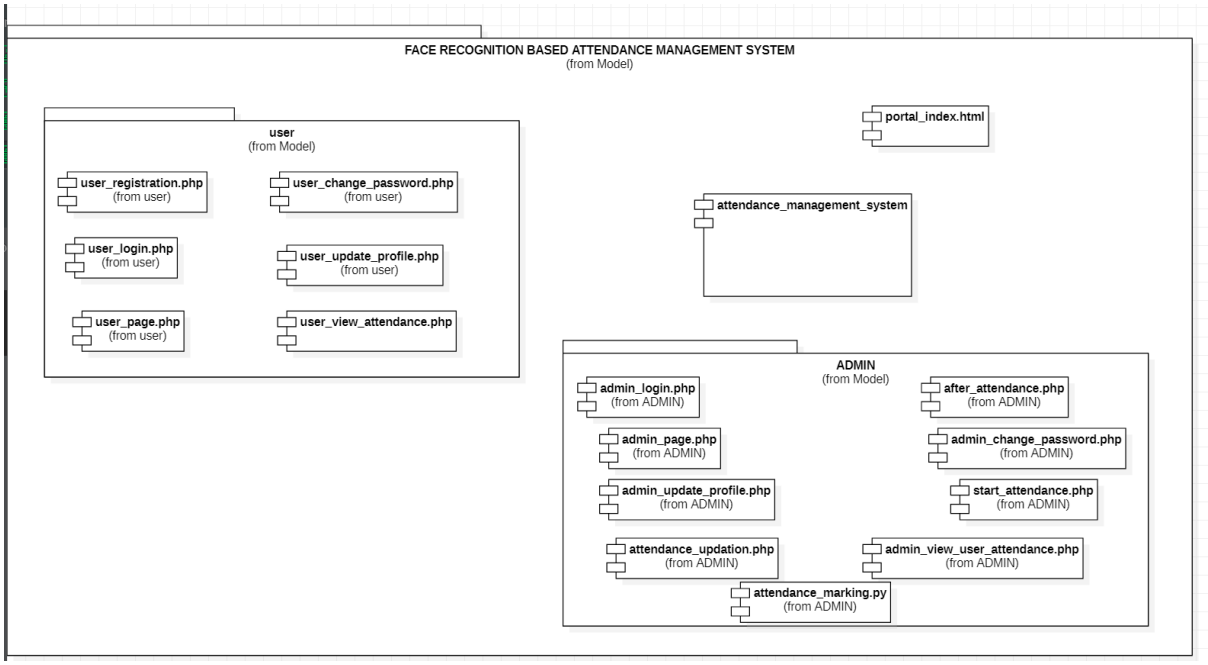
13. CONSTRUCTION OF IMPLEMENTATION DIAGRAMS

Component diagrams:

In a large project there will be many files that make up the system. These files will have dependencies on one another. The nature of these dependencies will depend on the language or languages used for the development and may exist at compile-time, at link-time or at run-time. There are also dependencies between source code files and the executable files or byte code files that are derived from them by compilation. Component diagrams are one of the two types of implementation diagram in UML. Component diagrams show these dependencies between software components in the system. Stereotypes can be used to show dependencies that are specific to particular languages also.

A component diagram shows the allocation of classes and objects to components in the physical design of a system. A component diagram may represent all or part of the component architecture of a system along with dependency relationships.

COMPONENT DIAGRAM FOR FRAMS



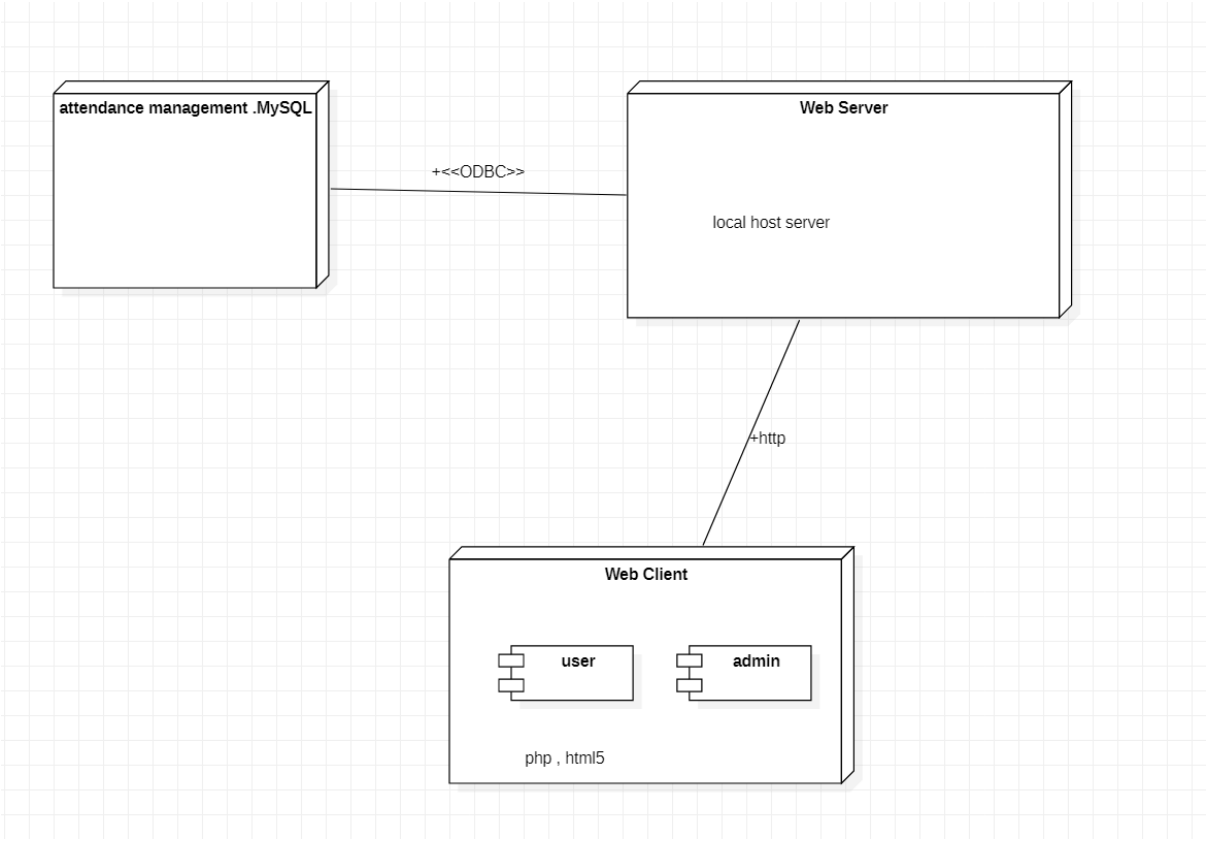
Deployment diagrams:

The second type of implementation diagram provided by UML is the deployment diagram. Deployment diagrams are used to show the configuration of run-time processing elements and the software components and processes that are located on them.

Deployment diagrams are made up of nodes and communication associations. Nodes are typically used to show computers and the communication associations show the network and protocols that are used to communicate between nodes. Nodes can be used to show other processing resources such as people or mechanical resources.

Nodes are drawn as 3D views of cubes or rectangular prisms, and the following figure shows a simplest deployment diagram where the nodes connected by communication associations.

DEPLOYEMENT DIAGRAM FOR FRAMS



14. SAMPLE APPLICATION CODE AND DATABASE

Various modules in the system are

1.Portal Index
2.User Registration
3.User Login
4.User View Profile
5.User Change Password
6.User Update Profile
7.User View Attendance
8.Admin Login
9.Admin View Profile
10.Admin Change Password
11.Admin Update Profile
12.Admin View User Attendance
13.Admin Mark Attendance
14.Start Attendance
15.After Attendance

CODE FOR USER REGISTRATION :

```php
<?php
session_start();

$dbhost="localhost";
$dbuser="root";
$dbpass='';
$dbname="attendance_management";
$con = mysqli_connect($dbhost,$dbuser,$dbpass,$dbname);

if(!$con)
{
    die("failed to connect");
}

include("functions.php");
if($_SERVER['REQUEST_METHOD']== "POST")
{
    $user_name = $_POST['user_name'];
    $full_name = $_POST['full_name'];
    $email_id =  $_POST['email_id'];
    $phone_num = $_POST['phone_number'];
    $date_of_birth = $_POST['date_of_birth'];
    $image = $_FILES['upload_photo']['name'];
    $tmpname = $_FILES['upload_photo']['tmp_name'];
    $folder = "attendance_images/";
    move_uploaded_file($tmpname,$folder.$image);
    $password = $_POST['password'];
    $temp_img_name = $image;
    $temp_img_name = trim($temp_img_name,".jpg");
```

```
    $temp_img_name = strtolower($temp_img_name);
    $temp_user_name = strtolower($user_name);
    $user_names = [];

    if(!empty($_POST['user_name']) && !empty($_POST['password']) &&
!is_numeric($user_name))
    {
        $query_get_user_names = "select email_id from users";
        $get_user_names = mysqli_query($con,$query_get_user_names);
        while ($row = mysqli_fetch_assoc($get_user_names)) {
            $user_names[] =  $row['email_id'];
        }
        $user_name_flag = 0;
        foreach($user_names as $value){
            if(strcmp($value,$email_id)==0){
                $user_name_flag = 1;
                break;
            }
        }
        if($user_name_flag==0){
            if(strcmp($temp_user_name,$temp_img_name)==0){
                $user_id = random_num(20);
                $query = "insert into
users(user_id,user_name,full_name,email_id,password,phone_no,image,date
_of_birth) values
('$user_id','$user_name','$full_name','$email_id','$password','$phone_n
um','$image','$date_of_birth')";

                mysqli_query($con,$query);
                $query_id = "select id from users where
user_id=$user_id";
                $query_id_result = mysqli_query($con,$query_id);
                $id_value_query = mysqli_fetch_assoc($query_id_result);
                $id_value= $id_value_query['id'];
                $query_attendance_details = "insert into
attendance_details(user_id) values('$id_value')";
                mysqli_query($con,$query_attendance_details);
                echo"<script>alert('YOU HAVE REGISTERED
SUCCESSFULLY')</script>";

                header("Location: user_login.php");
                die;
            }
            else{
                echo "<script>alert('please make sure that username and
photo file are same!!');</SCRIPT>";
            }

        }
        else{
            echo "<script>alert('YOU HAVE ALREADY
REGISTERED');</SCRIPT>";
        }

    }
    else{
        echo "<script>alert('PLEASE ENTER ALL DETAILS')</script>";
```

```
    }
}
?>

<!DOCTYPE html>
<html>
    <head>
        <title>USER REGISTRATION</title>
        <link rel="stylesheet" type="text/css"
href="attendance_system_styles.css">
    </head>
    <body>
        <header class="heading">ATTENDANCE MANAGEMENT SYSTEM</header>
        <a class="home_class" href="portal_index.html">HOME</a>

        <div class="registration_form">
            <p class="register_form_heading">REGISTRATION FORM</p>
            <form class="form_class" method="post"
enctype="multipart/form-data">
                <input class="registration_inputs" type="text"
name="user_name" placeholder="USER NAME"><br><br><br>
                <input class="registration_inputs" type="text"
name="full_name" placeholder="FULL NAME"><br><br><br>
                <input class="registration_inputs" type="email"
name="email_id" placeholder="EMAIL ID"><br><br><br>
                <input class="registration_inputs" type="number"
name="phone_number" placeholder="PHONE NUMBER"><br><br><br>
                <p class="register_field_name">DATE OF BIRTH : </p>
                <br><input class="registration_inputs" type="date"
name="date_of_birth" placeholder="DATE OF BIRTH"><br><br><br>
                <p class="user_name_photo">PLEASE MAKE SURE THAT YOUR
USER NAME AND PHOTO FILE NAME ARE SAME</p><br>
                <p class="register_field_name">UPLOAD PHOTO : </p><br>
                <input class="registration_inputs" type="file"
name="upload_photo" value="upload photo" required placeholder="UPLOAD
PHOTO"><br><br><br>
                <input class="registration_inputs" type="password"
name="password" placeholder="PASSWORD"><br><br><br><br>
                <input class="register_button" type="submit"
name="submit" value="REGISTER">
            </form>
        </div>
    </body>
</html>
```

CODE FOR USER VIEW PROFILE :

```
<?php
session_start();
    $dbhost="localhost";
    $dbuser="root";
    $dbpass='';
    $dbname="attendance_management";
    $con = mysqli_connect($dbhost,$dbuser,$dbpass,$dbname);
```

```
    if(!$con)
    {
        die("failed to connect");
    }
    include("functions.php");
    $user_data = check_user_login($con);

?>

<!DOCTYPE html>
<html>
    <head>
        <title>USER</title>
        <link rel="stylesheet" type="text/css"
href="attendance_system_styles.css">
    </head>
    <body>
        <header class="heading">ATTENDANCE MANAGEMENT SYSTEM</header>
        <div id="navpage" class="sidemenupage">
            <a href="user_update_profile.php">UPDATE PROFILE</a>
            <a href="user_change_password.php">CHANGE PASSWORD</a>
            <a href="user_view_attendance_report.php">VIEW
ATTENDANCE</a>
        </div>

        <?php
            $id= $user_data['id'];
            $query_user_details = "select * from users where id = $id";
            $get_user_details = mysqli_query($con,$query_user_details);
            $user_details = mysqli_fetch_assoc($get_user_details);

        ?>

        <a class="logout_class" href="logout.php">Logout</a>

        <div class="profile_display">
            <h1 class="profile_heading">USER PROFILE</h1>
            <table border="2" class="user_profile_table"
cellspacing="0px">
                <tr class="row_img" align=center><td colspan="2"
border="0px" align=center>
                        <img class="user_photo" src="<?php echo
"attendance_images/".$user_details['image']?>" ></td>
                </tr>
                <tr>
                    <td class="field_name">USER NAME</td>
                    <td class="field_value"><?php echo
$user_details['user_name']?></td>
                </tr>
                <tr>
                    <td class="field_name">FULL NAME</td>
                    <td class="field_value"><?php echo
$user_details['full_name']?></td>
                </tr>
                <tr>
                    <td class="field_name">EMAIL ID</td>
                    <td class="field_value"><?php echo
```

```
$user_details['email_id']?></td>
                </tr>
                <tr>
                    <td class="field_name">PHONE NUMBER</td>
                    <td class="field_value"><?php echo
$user_details['phone_no']?></td>
                </tr>
                <tr>
                    <td class="field_name">DATE OF BIRTH</td>
                    <td class="field_value"><?php echo
$user_details['date_of_birth']?></td>
                </tr>
            </table>
        </div>

    </body>
</html>
```

CODE FOR USER CHANGE PASSWORD :

```
<?php
session_start();

$dbhost="localhost";
$dbuser="root";
$dbpass='';
$dbname="attendance_management";
$con = mysqli_connect($dbhost,$dbuser,$dbpass,$dbname);

if(!$con)
{
    die("failed to connect");
}

include("functions.php");
$user_data = check_user_login($con);
$user_id = $user_data['id'];
if($_SERVER['REQUEST_METHOD']== "POST")
{
    $previous_password =  $_POST['previous_password'];
    $new_password = $_POST['new_password'];
    $confirm_password = $_POST['confirm_password'];

    if(!empty($previous_password) && !empty($new_password) &&
!empty($confirm_password)){
        if($previous_password==$user_data['password']){
            echo "loop";
            if($new_password == $confirm_password){
                echo "entered";
                $query_update_password = "update users set password =
'$new_password' where id= $user_id";
                mysqli_query($con,$query_update_password);
            }
```

```
            else{
                echo "<script>alert('please confirm password
correctly')</script>";
            }
        }
        else{
            echo "<script>alert('please enter previous password
correclty')</script>";
        }
    }
    else{
        echo "<script>alert('PLEASE ENTER ALL DETAILS')</script>";
    }
}
?>

<!DOCTYPE html>
<html>
    <head>
        <title>CHANGE PASSWORD</title>
        <link rel="stylesheet" type="text/css"
href="attendance_system_styles.css">
    </head>
    <body>
        <header class="heading">ATTENDANCE MANAGEMENT SYSTEM</header>
        <div id="navpage" class="sidemenupage">
                <a href="user_page.php">VIEW PROFILE</a>
                <a href="user_update_profile.php">UPADTE PROFILE</a>
                <a href="user_view_attendance_report.php">VIEW
ATTENDANCE</a>
        </div>
        <a class="logout_class" href="logout.php">Logout</a>
        <div class="change_password_form">
            <p class="change_password_heading">CHANGE PASSWORD</p>
            <form class="form_class" method="post">
                <input class="change_password_inputs" type="password"
name="previous_password" placeholder="PREVIOUS PASSWORD"
required><br><br>
                <input class="change_password_inputs" type="password"
name="new_password" placeholder="NEW PASSWORD" required><br><br>
                <input class="change_password_inputs" type="password"
name="confirm_password" placeholder="CONFIRM PASSWORD"
required><br><br>
                <input class="change_password_button" type="submit"
name="submit" value="CHANGE PASSWORD">
            </form>
        </div>


    </body>
</html>
```

CODE FOR ADMIN UPDATE PROFILE :

```
<?php
session_start();

$dbhost="localhost";
$dbuser="root";
$dbpass='';
$dbname="attendance_management";
$con = mysqli_connect($dbhost,$dbuser,$dbpass,$dbname);

if(!$con)
{
    die("failed to connect");
}

include("functions.php");
$admin_data = check_admin_login($con);
$admin_id = $admin_data['id'];
if($_SERVER['REQUEST_METHOD']== "POST")
{
    $new_email_id =  $_POST['email_id'];
    $new_phone_num = $_POST['phone_number'];
    $new_date_of_birth = $_POST['date_of_birth'];

    if(!empty($new_phone_num)){
        $query_update_phone_num = "update admins set phone_no =
'$new_phone_num' where id = $admin_id";
        mysqli_query($con,$query_update_phone_num);
        echo "<script>alert('UPDATED SUCCESSFULLY!!')</script>";
    }
    if(!empty($new_date_of_birth)){
        $query_update_date_of_birth = "update admins set date_of_birth
= '$new_date_of_birth' where id = $admin_id";
        mysqli_query($con,$query_update_date_of_birth);
        echo "<script>alert('UPDATED SUCCESSFULLY!!')</script>";
    }
    if(!empty($new_email_id)){
        $query_update_email_id = "update admins set email_id =
'$new_email_id' where id = $admin_id";
        mysqli_query($con,$query_update_email_id);
        echo "<script>alert('UPDATED SUCCESSFULLY!!')</script>";
    }
}
?>


<!DOCTYPE html>
<html>
    <head>
        <title>UPDATE PROFILE</title>
        <link rel="stylesheet" type="text/css"
href="attendance_system_styles.css">
    </head>
    <body>
```

```
        <header class="heading">ATTENDANCE MANAGEMENT SYSTEM</header>
        <div id="navpage" class="sidemenupage">
            <a href="admin_page.php">VIEW PROFILE</a>
            <a href="admin_change_password.php">CHANGE PASSWORD</a>
            <a href="admin_view_user_attendance.php">VIEW USER
ATTENDANCE</a>
            <a href="start_attendance.html">MARK ATTENDANCE</a>
        </div>
        <a class="logout_class" href="logout.php">Logout</a>
        <div class="profile_update_form">
            <p class="update_profile_heading">UPDATE PROFILE</p>
            <form method="post">
                <input class="update_profile_inputs" type="email"
name="email_id" placeholder="EMAIL ID"><br><br>
                <input class="update_profile_inputs" type="number"
name="phone_number" placeholder="PHONE NUMBER"><br><br>
                <input class="update_profile_inputs" type="date"
name="date_of_birth" placeholder="DATE OF BIRTH"><br><br>
                <input class="update_profile_button" type="submit"
name="submit" value="UPDATE">
            </form>
        </div>
    </body>
</html>
```

CODE FOR ADMIN VIEW USER ATTENDANCE :

```
<?php
session_start();

$dbhost="localhost";
$dbuser="root";
$dbpass='';
$dbname="attendance_management";
$con = mysqli_connect($dbhost,$dbuser,$dbpass,$dbname);

if(!$con)
{
    die("failed to connect");
}

include("functions.php");
$admin_data = check_admin_login($con);

$myarray=[];
$my_array=[];
$date_array=[];
$user_id =0;
$date_falg =0;
$percentage=0;
$system_end_date=0;
$system_start_date=0;
$query_1 = 'select * from attendance_details';
$result_1 = mysqli_query($con,$query_1);
```

```
$i = 0;
$j =0;
while ($j < mysqli_num_fields($result_1))
{
    $fld = mysqli_fetch_field_direct($result_1, $j);
    $my_array[]=$fld->name;
    $j = $j + 1;
}
$system_start_date = $my_array[1];
$system_end_date = $my_array[$j-1];

if ($_SERVER['REQUEST_METHOD'] == "POST") {

    if(!empty($_POST['user_name'])){
        $user_name = $_POST['user_name'];
        $user_name="'".$user_name."'";
        $query_get_user_id = "select `id` from users where user_name =
$user_name";
        $get_user_id = mysqli_query($con,$query_get_user_id);
        $user_id_get = mysqli_fetch_assoc($get_user_id);
        $user_id = $user_id_get['id'];
        $percentage = calculate_percentage($con,$user_id);
        $query_user_details = "select * from users where id =
$user_id";
        $get_user_details = mysqli_query($con,$query_user_details);
        $user_details = mysqli_fetch_assoc($get_user_details);
        $user_full_name = $user_details['full_name'];
        $user_email_id = $user_details['email_id'];
        $user_phone_num = $user_details['phone_no'];
        $user_image = $user_details['image'];
        $user_date_of_birth = $user_details['date_of_birth'];
        if(!empty($_POST["report"])){
            $opt_selected = $_POST["report"];
            if($opt_selected == "view detailed report"){
                $query = 'select * from attendance_details';
                $result = mysqli_query($con,$query);
                $i = 0;
                while ($i < mysqli_num_fields($result))
                {
                    $fld = mysqli_fetch_field_direct($result, $i);
                    $myarray[]=$fld->name;
                    $i = $i + 1;
                }
                foreach($myarray as $value){
                    $date_array[]=$value;
                }
            }
            else if($opt_selected=="view date wise report"){
                if(!empty($_POST['start_date'])){
                    if(!empty($_POST['end_date'])){
                        $start_date = $_POST['start_date'];
                        $end_date = $_POST['end_date'];
                        $start_date= strval($start_date);
                        $end_date = strval($end_date);
                        $start_date = "'".$start_date."'";
                        $end_date = "'".$end_date."'";
```

```
                        $query = 'select * from attendance_details';
                        $result = mysqli_query($con,$query);
                        $i = 0;
                        while ($i < mysqli_num_fields($result))
                        {
                            $fld = mysqli_fetch_field_direct($result,
$i);
                            $myarray[]=$fld->name;
                            $i = $i + 1;
                        }

                        foreach($myarray as $value){
                            if((strcmp($value,$start_date)>=0) &&
(strcmp($value,$end_date)<=0)){
                                $date_array[]=$value;
                            }

                        }
                    }
                    else{
                        echo "<script>alert('PLEASE SPECIFY END
DATE')</script>";
                    }

                }
                else{
                    echo"<script>alert('PLEASE SPECIFY START
DATE')</script>";
                }
            }

        }
        else{
            echo"<script>alert('PLEASE SELECT AN OPTION')</script>";
        }
    }
    else{
        echo "<script>alert('PLEASE SELECT USER NAME')</script>";
    }

}
?>

<!DOCTYPE html>
<html>
    <head>
        <title>VIEW USER ATTENDANCE</title>
        <link rel="stylesheet" type="text/css"
href="attendance_system_styles.css">
        <script>
            function enter_dates()
            {
                if(document.getElementById('date').checked){

document.getElementById("dates_input").style.visibility="visible";
                }
                else{
```

```
document.getElementById("dates_input").style.visibility="visible";
                }
            }
        </script>
    </head>
    <body>
        <header class="heading">ATTENDANCE MANAGEMENT SYSTEM</header>
        <div id="navpage" class="sidemenupage">
            <a href="admin_update_profile.php">UPDATE PROFILE</a>
            <a href="admin_change_password.php">CHANGE PASSWORD</a>
            <a href="admin_view_user_attendance.php">VIEW USER
ATTENDANCE</a>
            <a href="start_attendance.html">MARK ATTENDANCE</a>
        </div>
        <a class="logout_class" href="logout.php">Logout</a>
        <div class="view_block">
            <p class="view_form_heading">VIEW USER ATTENDANCE</p>
            <div class="date_form">
                <form method="post" class="form_class">
                    USER NAME :
                    <select class="view_inputs" name="user_name">
                        <option disabled selected>-- Select user --
</option>
                        <?php

                        $records = mysqli_query($con, "SELECT user_name
from users");

                        while($data = mysqli_fetch_array($records))
                        {
                            echo "<option value='". $data['user_name']
."'>" .$data['user_name'] ."</option>";
                        }
                        ?>

                    </select><br>
                    <input  class="radio_buttons_1" type="radio"
id="normal" name="report" value="view detailed report">
                    <label class="dates_label" for="normal">view
detailed report</label><br>
                    <input class="radio_buttons_1" type="radio"
id="date" name="report" value="view date wise report"
onclick="javascript:enter_dates();">
                    <label class="dates_label" for="date">view date
wise report</label><br>
                    <input class="dates_button" type="submit"
name="submit" value="view" onClick="location.href='#profile_display'">
                    <div id = "dates_input" STYLE="visibility: hidden">
                        START DATE : <input class="view_inputs"
type="date" id="start_date" name="start_date"><br>
                        END DATE : <input class="view_inputs"
type="date" id="end_date" name="end_date">
                        <p class="dates_warning">PLEASE SELECT DATES
BETWEEN <?php echo $system_start_date?> AND <?php echo $system_end_date
?> (BOTH INCLUSIVE)</p>
                    </div>
```

```
                </form>
            </div>
            <?php
                if(!empty($user_details))
                {
                    echo "<div class='view_profile_display'
id='profile_display'>";
                        echo "<table border=2
class='view_profile_table'><tr class='row_img'><td colspan='2'
align='center'><img class='user_photo'
src='attendance_images/".$user_image."'</td></tr>";
                        echo "<tr><td class='field_name'>USER
NAME</td><td class='field_value'>".$user_name."</td></tr>";
                        echo "<tr><td class='field_name'>FULL
NAME</td><td class='field_value'>".$user_full_name."</td></tr>";
                        echo "<tr><td class='field_name'>EMAIL
ID</td><td class='field_value'>".$user_email_id."</td></tr>";
                        echo "<tr><td class='field_name'>PHONE
NUMBER</td><td class='field_value'>".$user_phone_num."</td></tr>";
                        echo "<tr><td class='field_name'>DATE OF
BIRTH</td><td
class='field_value'>".$user_date_of_birth."</td></tr></table></div>";
                }
                echo "<p class='attendance_percentage_class'>ATTENDANCE
PERCENTAGE = ".$percentage."</p>";
            ?>

            <table border="2" class="attendance_table">
                <tr>
                    <th class="field_name">date</th>
                    <th class="field_name">morning_session</th>
                    <th class="field_name">evening_session</th>
                </tr>
                <?php
                    foreach($date_array as $value){

                        if($value!="USER_ID") {
                            echo "<tr><td class='field_value'>" .
$value . "</td>";
                            $query_attendance_details = "select
`$value` from attendance_details where user_id = $user_id";
                            $details =
mysqli_query($con,$query_attendance_details);
                            $attendance_details =
mysqli_fetch_assoc($details);
                            $attendance_value =
$attendance_details[$value];
                            if($attendance_value=='5'){
                                echo "<td
class='field_value'>".'present'."</td>";
                                echo "<td
class='field_value'>".'present'."</td></tr>";
                            }
                            else if($attendance_value=='3'){
                                echo "<td
class='field_value'>".'absent'."</td>";
                                echo "<td
```

```
class='field_value'>".'present'."</td></tr>";
                                }
                                else if($attendance_value=='1'){
                                    echo "<td
class='field_value'>".'present'."</td>";
                                    echo "<td
class='field_value'>".'absent'."</td></tr>";
                                }
                                else if($attendance_value=='0'){
                                    echo "<td
class='field_value'>".'absent'."</td>";
                                    echo "<td
class='field_value'>".'absent'."</td></tr>";
                                }

                            }
                        }

                    ?>
                </table>
            </div>
        </body>
</html>
```

CODE FOR ATTENDANCE MARKING :

```
import cv2
import numpy as np
import face_recognition
import os
import time
from datetime import date
import mysql.connector
import mysql
try:
    connection = mysql.connector.connect(host='localhost',
                                         database='attendance_management',
                                         user='root',
                                         password='')
    cursor = connection.cursor()
except mysql.connector.Error as error:
    connection.rollback()

path = 'attendance_images'
images = []
classnames = []
myList = os.listdir(path)

for cl in myList:
    curImg = cv2.imread(f'{path}/{cl}')
    images.append(curImg)
    classnames.append(os.path.splitext(cl)[0])
```

```
def findEncodings(images):
    encodeList = []
    for img in images:
        img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
        encode = face_recognition.face_encodings(img)[0]
        encodeList.append(encode)
    return encodeList


encodeListKnown = findEncodings(images)

start = time.time()
period = 20

cap = cv2.VideoCapture(0)
attendance_list=[]
while True:
    success , img = cap.read()
    imgS = cv2.resize(img,(0,0),None,0.25,0.25)
    imgS = cv2.cvtColor(imgS,cv2.COLOR_BGR2RGB)
    mark_date = date.today()
    facesCurFrame = face_recognition.face_locations(imgS)
    encodesCurFrame =
face_recognition.face_encodings(imgS,facesCurFrame)

    for encodeFace,faceLoc in zip(encodesCurFrame,facesCurFrame):
        matches =
face_recognition.compare_faces(encodeListKnown,encodeFace)
        faceDis =
face_recognition.face_distance(encodeListKnown,encodeFace)
        matchIndex = np.argmin(faceDis)

        if matches[matchIndex]:
            name = classnames[matchIndex].upper()
            #print(name)
            y1,x2,y2,x1 = faceLoc
            y1,x2,y2,x1 = y1*4,x2*4,y2*4,x1*4
            cv2.rectangle(img,(x1,y1),(x2,y2),(0,255,0))
            cv2.rectangle(img,(x1,y2-35),(x2,y2),(0,255,0),cv2.FILLED)
            cv2.putText(img,name,(x1+6,y2-
6),cv2.FONT_HERSHEY_COMPLEX,1,(255,255,255),2)

            query_get_id = """select id from users where user_name =
%s"""
            cursor.execute(query_get_id,(name,))
            id_list = [i[0] for i in cursor.fetchall()]
            if(id_list):
                attendance_list.append(id_list[0])

    cv2.imshow('webcam',img)
    if time.time() > start + period:
        break
    if cv2.waitKey(100) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

```
if connection.is_connected():
    cursor.close()
    connection.close()
attendance_list = list(set(attendance_list))
print(attendance_list)
```

Database tables created in application

ADMIN TABLE :

ADMIN Table stores details of admins.

Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	*NULL*	auto_increment
full_name	varchar(100)	NO	MUL	*NULL*	
email_id	varchar(75)	NO	MUL	*NULL*	
password	varchar(25)	NO		*NULL*	
phone_no	bigint(10)	NO	MUL	*NULL*	
date_of_birth	datetime	NO	MUL	*NULL*	

USERS TABLE:

USERS Table store the details of users.

Field	Type	Null	Key	Default	Extra
id	bigint(20)	NO	PRI	*NULL*	auto_increment
user_id	bigint(20)	NO		*NULL*	
user_name	varchar(100)	NO	MUL	*NULL*	
full_name	varchar(100)	NO	MUL	*NULL*	
email_id	varchar(75)	NO	MUL	*NULL*	
password	varchar(25)	NO		*NULL*	
phone_no	varchar(11)	NO	MUL	*NULL*	
image	text	NO		*NULL*	
date_of_birth	datetime	NO	MUL	*NULL*	

ATTENDANCE_DETAILS TABLE :

ATTENDANCDE_DETAILS store the attendance marked for each user for each working day. A column is created with date as column name. The ‘0’ value in each cell represents that attendance is not yet marked for that user on that particular day.
The value ‘1’ in each cell represents that attendance is only marked only in morning session for that user on that particular day. The value ‘3’ in each cell represents that attendance is only marked in evening session for that user on that particular day. The value ‘5’ in each cell represents that attendance is marked in both morning and evening sessions.

Field	Type	Null	Key	Default	Extra
USER_ID	bigint(20)	NO	PRI	*NULL*	
'2021-04-18'	varchar(20)	NO		0	
'2021-04-19'	varchar(20)	NO		0	
'2021-04-20'	varchar(20)	NO		0	
'2021-04-21'	varchar(20)	NO		0	
'2021-04-22'	varchar(20)	YES		0	

SESSION_DETAILS TABLE :

Session_details table helps to know whether attendance has taken or not in each session i.e, in morning session and afternoon session. The value ‘0’ in each cell indicates that attendance has not yet taken in that particular session. The value ‘1’ in each cell indicates that attendance has already been taken in that particular session**.**

Field	Type	Null	Key	Default	Extra
session_id	varchar(20)	NO	PRI	*NULL*	
'2021-04-18'	varchar(20)	NO		0	
'2021-04-19'	varchar(20)	NO		0	
'2021-04-20'	varchar(20)	NO		0	
'2021-04-21'	varchar(20)	NO		0	
'2021-04-22'	varchar(20)	NO		0	

15. TESTING

The main purpose of testing FRAMS is to ensure that all the activities and functionalities of this software run smoothly with no errors and it remains protected.

FOR ADMIN :

- Verify admin login with valid and invalid data
- Verify admin view profile
- Verify admin update profile with valid and invalid data
- Verify admin change password with valid and invalid data
- Verify admin view user attendance with valid and invalid data
- Verify admin mark attendance

FOR USER :

- Verify user registration with valid and invalid data.
- Verify user login with valid and invalid data
- Verify user view profile.
- Verify user update profile with valid and invalid data.
- Verify user change password with valid and invalid data.
- Verify user view attendance with valid and invalid data.

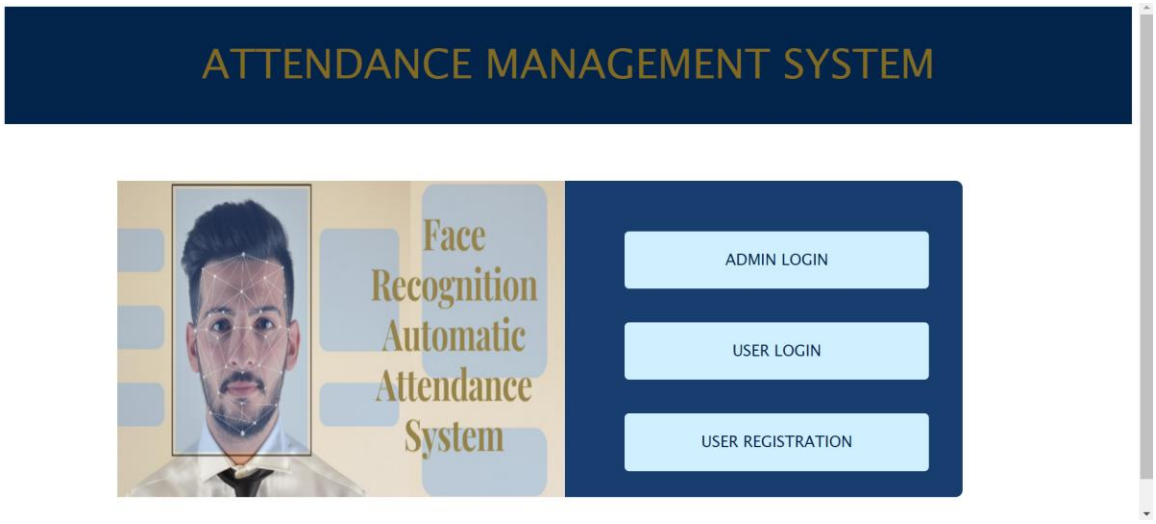
Test case no	Functionality to be checked	Actual input	Actual output	Expected output	Status
1	Verify admin login/verify user login	Given valid email id & valid password	Login success	Login success	Pass
		Given valid email id & invalid password	Deny login	Deny login	Pass
		Given invalid email id & valid password	Deny login	Deny login	Pass
		Given invalid email id & invalid password	Deny login	Deny login	Pass
2	Verify admin update profile/verify user update profile	Given new email id, new phone no, new date of birth	updated	updated	Pass
		Given only one field	updated	updated	Pass
		Given only two fields	updated	updated	Pass
3	Verify admin change password/ verify user change password	Given valid previous password, valid new password, correct confirm password	changed	Changed	Pass
		Given invalid previous password, valid new password, correct confirm password	Not changed	Not changed	Pass
		Given valid previous password, new password, wrong confirm password	Not changed	Not changed	Pass

4	Verify view user attendance / Verify user view attendance	Given invalid start date & invalid end date	Generates reports with warning	Deny request	fail
		Given valid start date & valid end date	Generates reports	Generates report	pass
		Given invalid start date & valid end date	Generates reports with warning	Deny request	fail
		Given valid start date & valid end date	Generates reports with warning	Deny request	fail
5	Verify mark attendance (mark attendance only once during morning session and evening session)	Click start attendance	Marks attendance numerous times but does not effect database	Marks only once each sessions	Partial full fillment

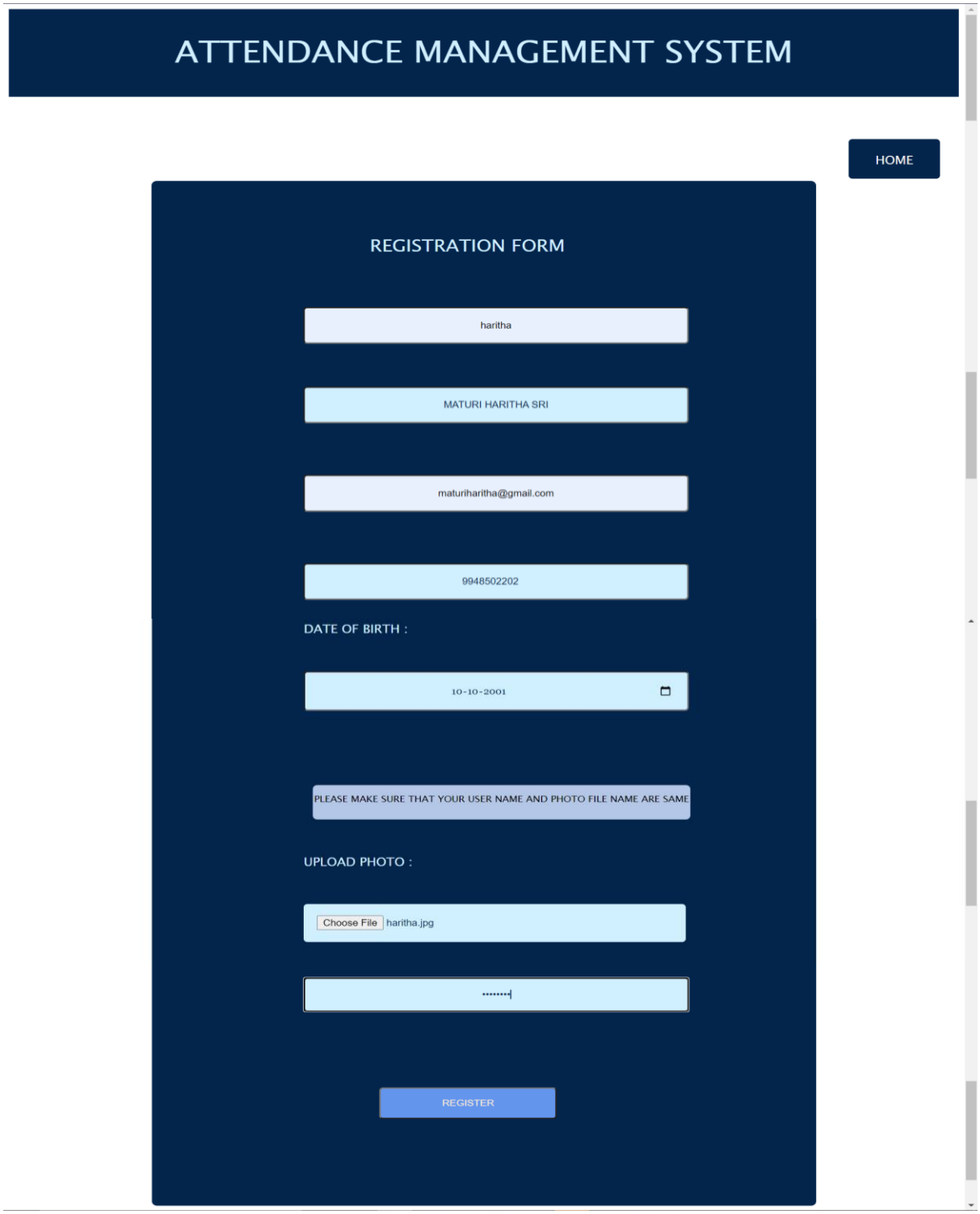
16. IMPLEMENTATION SCREEN SHOTS

The following are runtime GUI developed in the application along with functionality

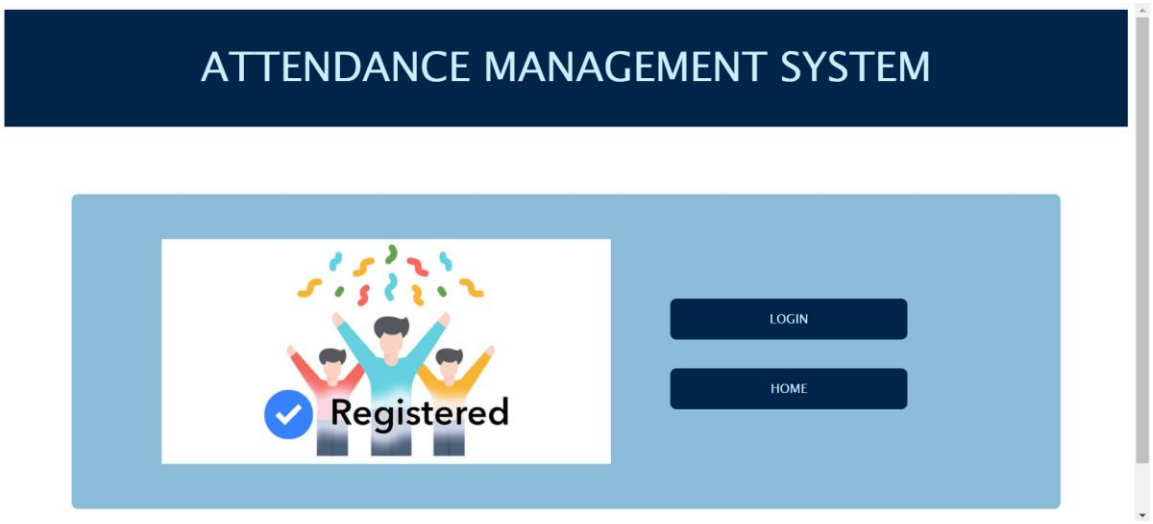
INDEX PAGE :



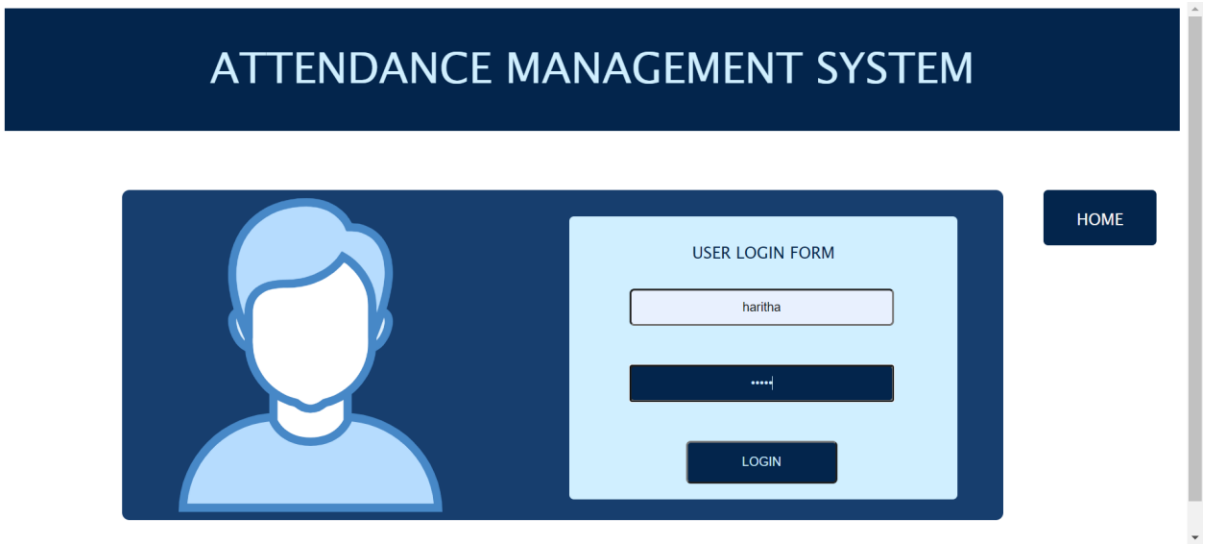
USER REGISTRATION :



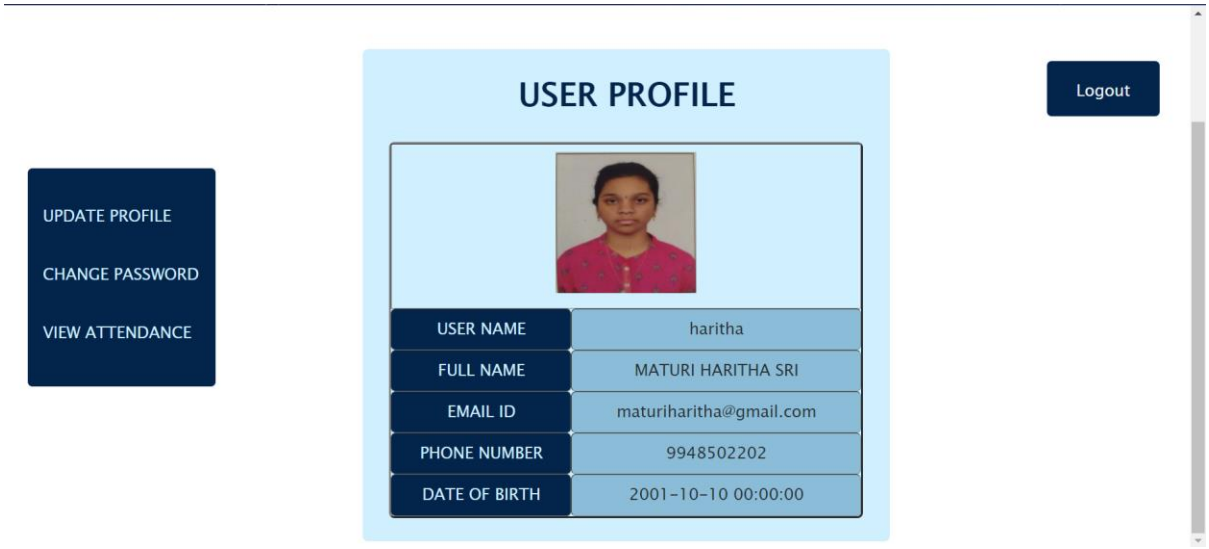
AFTER USER SUCCESSFUL REGISTRATION :



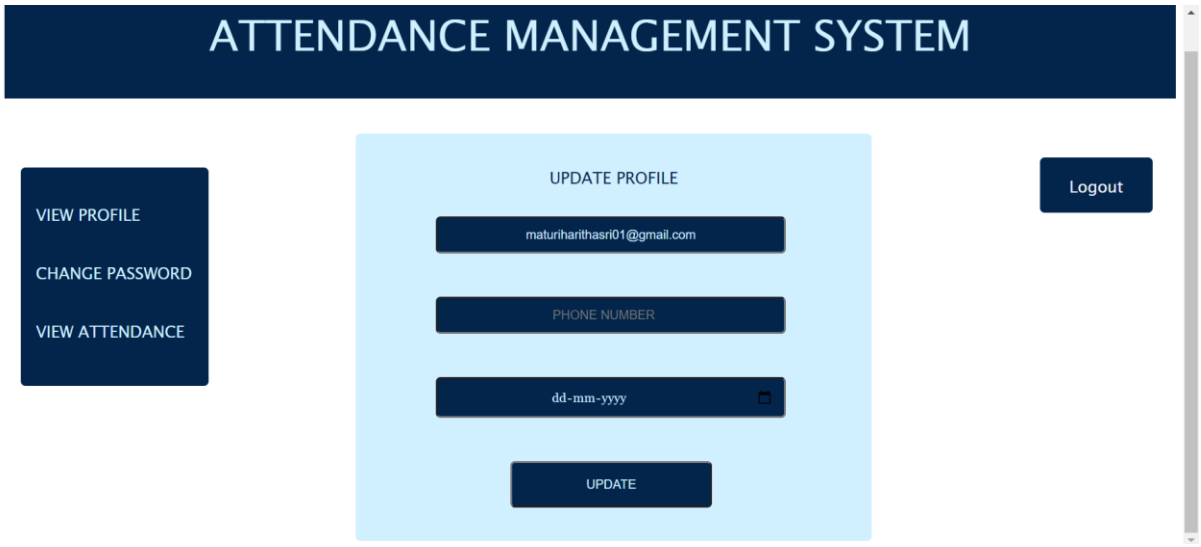
USER LOGIN :



USER VIEW PROFILE :



USER UPDATE PROFILE:



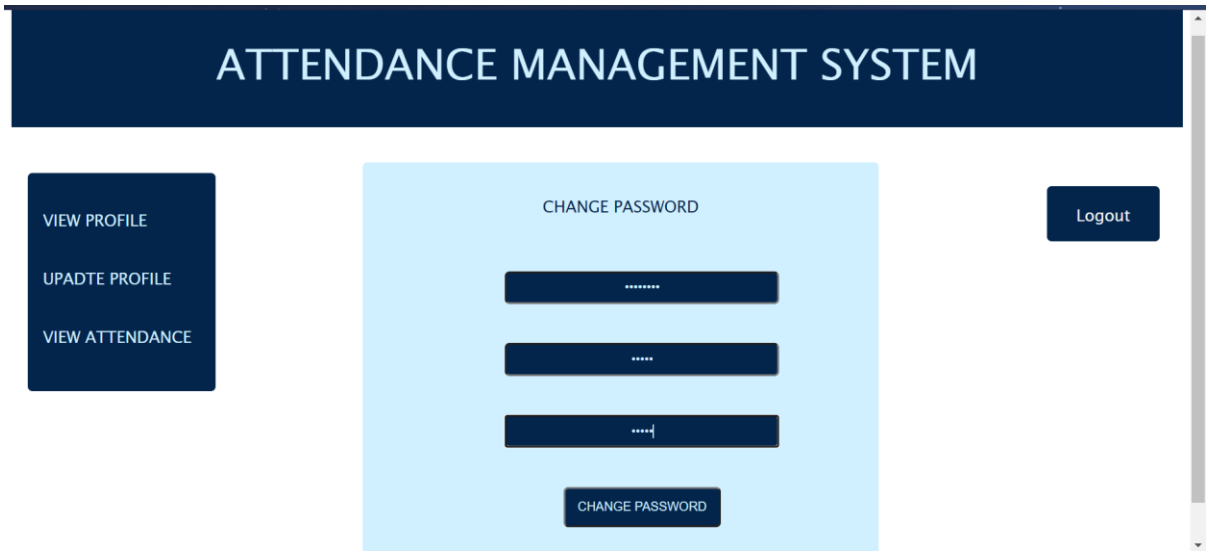
DATABASE BEFORE UPDATING PROFILE :

id	user_id	user_name	full_name	email_id	password	phone_no	image	date_of_birth
43	1	Bhavya	MATURI BHAVYA VENKATA NAGA SAI MANI	bhavyamaturi78592@gmail.com	bhavya6	6281919801	BHAVYA.jpg	2001-10-23 00:00:00
44	0	nimsy	DUDDU NIMSY	nimsiduddu@gmail.com	nimsy6	8520986627	Nimsy.jpeg	2001-08-27 00:00:00
48	7	madhuri	ADDANKI JAYA MADHURI	addankijayamadhuri11@gmail.com	madhuri6	6304600711	madhuri.jpeg	2001-06-10 00:00:00
52	8	haritha	MATURI HARITHA SRI	maturiharitha@gmail.com	haritha6	9948502202	haritha.jpg	2001-10-10 00:00:00
53	3	pravallika	MATURI VENKATA SAI PRAVALLIKA	pravallika@gmail.com	pravallika6	9182230015	PRAVALLIKA.jpg	2006-02-16 00:00:00
54	87	harshi	MALLEPULA HARSHITHA	harshitha.mallepula02@gmail.com	harshitha6	6302665510	harshi.jpeg	2002-04-11 00:00:00

DATABASE AFTER UPDATING PROFILE :

id	user_id	user_name	full_name	email_id	password	phone_no	image	date_of_birth
43	1	Bhavya	MATURI BHAVYA VENKATA NAGA SAI MANI	bhavyamaturi78592@gmail.com	bhavya6	6281919801	BHAVYA.jpg	2001-10-23 00:00:00
44	0	nimsy	DUDDU NIMSY	nimsiduddu@gmail.com	nimsy6	8520986627	Nimsy.jpeg	2001-08-27 00:00:00
48	7	madhuri	ADDANKI JAYA MADHURI	addankijayamadhuri11@gmail.com	madhuri6	6304600711	madhuri.jpeg	2001-06-10 00:00:00
52	8	haritha	MATURI HARITHA SRI	maturiharithasri01@gmail.com	haritha6	9948502202	haritha.jpg	2001-10-10 00:00:00
53	3	pravallika	MATURI VENKATA SAI PRAVALLIKA	pravallika@gmail.com	pravallika6	9182230015	PRAVALLIKA.jpg	2006-02-16 00:00:00
54	87	harshi	MALLEPULA HARSHITHA	harshitha.mallepula02@gmail.com	harshitha6	6302665510	harshi.jpeg	2002-04-11 00:00:00

USER CHANGE PASSWORD :



DATABASE BEFORE CHANGING PASSWORD :

id	user_id	user_name	full_name	email_id	password	phone_no	image	date_of_birth
43	1	Bhavya	MATURI BHAVYA VENKATA NAGA SAI MANI	bhavyamaturi78592@gmail.com	bhavya6	6281919801	BHAVYA.jpg	2001-10-23 00:00:00
44	0	nimsy	DUDDU NIMSY	nimsiduddu@gmail.com	nimsy6	8520986627	Nimsy.jpeg	2001-08-27 00:00:00
48	7	madhuri	ADDANKI JAYA MADHURI	addankijayamadhuri11@gmail.com	madhuri6	6304600711	madhuri.jpeg	2001-06-10 00:00:00
52	8	haritha	MATURI HARITHA SRI	maturiharithasri01@gmail.com	haritha6	9948502202	haritha.jpg	2001-10-10 00:00:00
53	3	pravallika	MATURI VENKATA SAI PRAVALLIKA	pravallika@gmail.com	pravallika6	9182230015	PRAVALLIKA.jpg	2006-02-16 00:00:00
54	87	harshi	MALLEPULA HARSHITHA	harshitha.mallepula02@gmail.com	harshitha6	6302665510	harshi.jpeg	2002-04-11 00:00:00

DATABASE AFTER CHANGING PASSWORD :

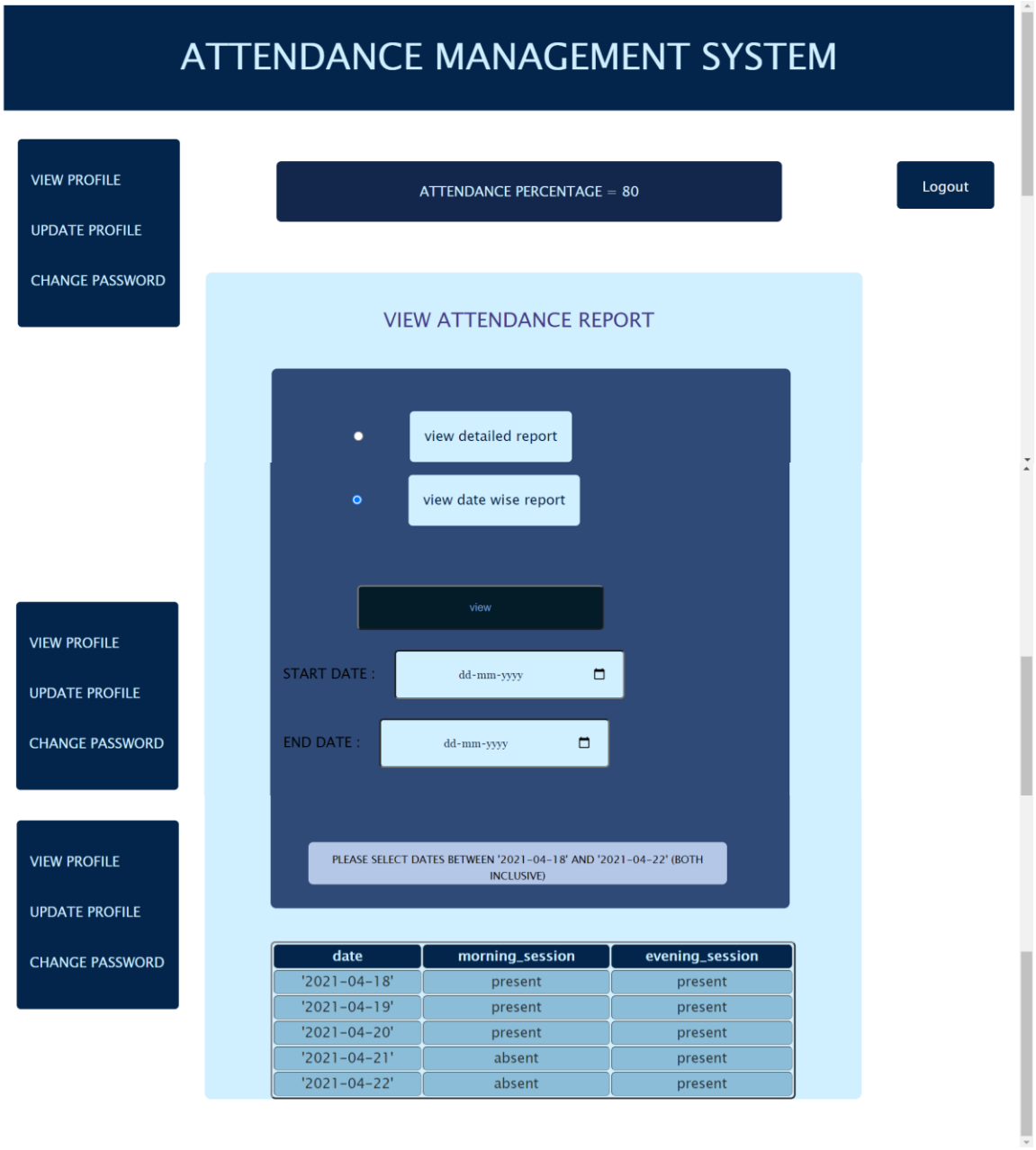
id	user_id	user_name	full_name	email_id	password	phone_no	image	date_of_birth
43	1	Bhavya	MATURI BHAVYA VENKATA NAGA SAI MANI	bhavyamaturi78592@gmail.com	bhavya6	6281919801	BHAVYA.jpg	2001-10-23 00:00:00
44	0	nimsy	DUDDU NIMSY	nimsiduddu@gmail.com	nimsy6	8520986627	Nimsy.jpeg	2001-08-27 00:00:00
48	7	madhuri	ADDANKI JAYA MADHURI	addankijayamadhuri11@gmail.com	madhuri6	6304600711	madhuri.jpeg	2001-06-10 00:00:00
52	8	haritha	MATURI HARITHA SRI	maturiharithasri01@gmail.com	hari6	9948502202	haritha.jpg	2001-10-10 00:00:00
53	3	pravallika	MATURI VENKATA SAI PRAVALLIKA	pravallika@gmail.com	pravallika6	9182230015	PRAVALLIKA.jpg	2006-02-16 00:00:00
54	87	harshi	MALLEPULA HARSHITHA	harshitha.mallepula02@gmail.com	harshitha6	6302665510	harshi.jpeg	2002-04-11 00:00:00

AFTER USER CHANGING PASSWORD :

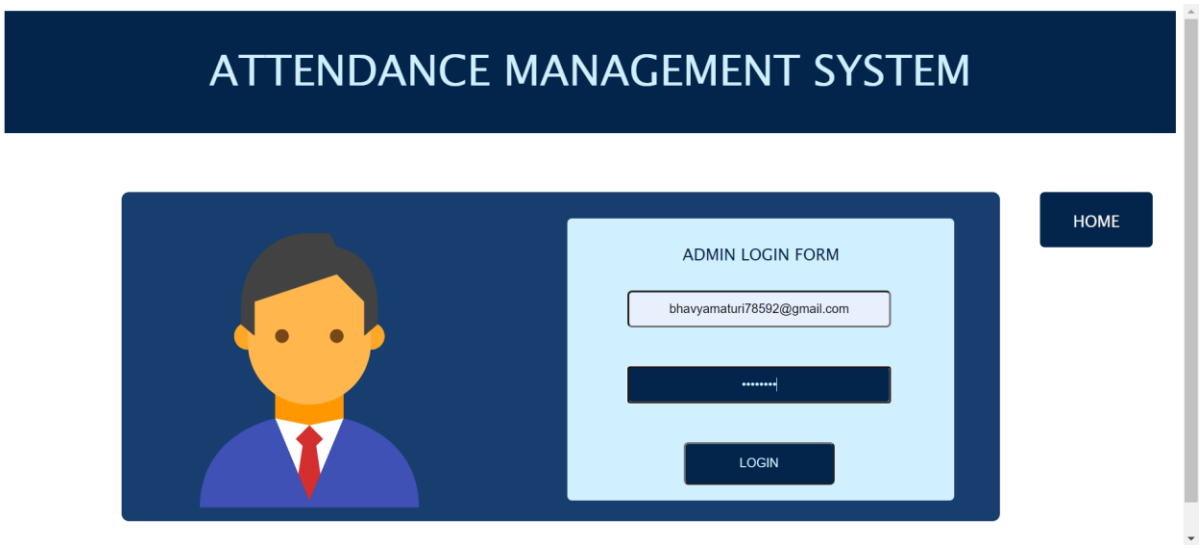
USER VIEW ATTENDANCE :

FOR DETAILED REPORT :

FOR DATEWISE REPORT :

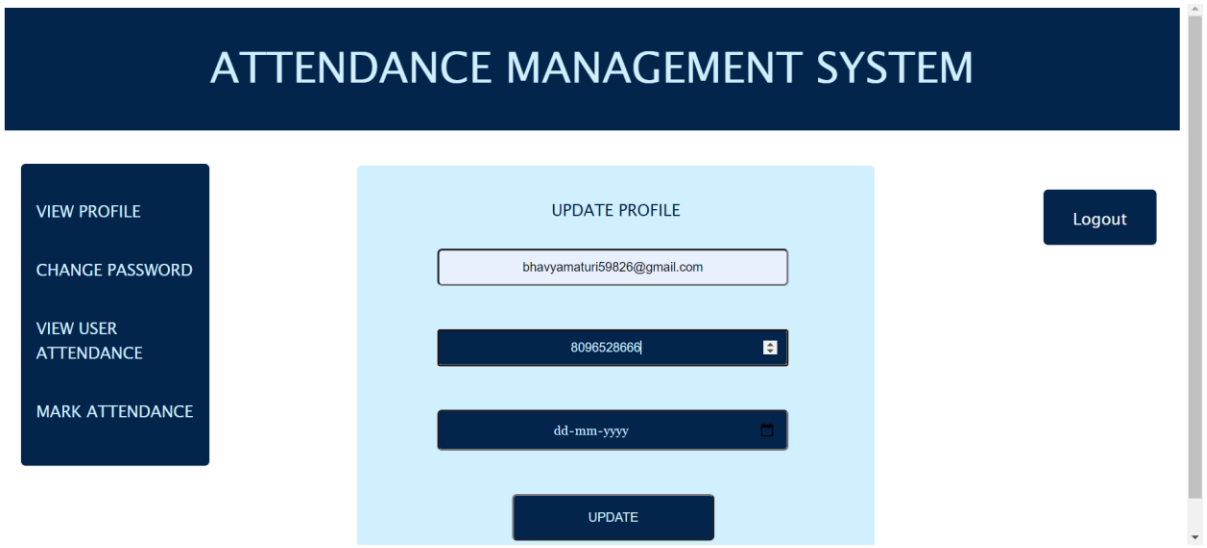


ADMIN LOGIN :



ADMIN VIEW PROFILE :

ADMIN UPDATE PROFILE :



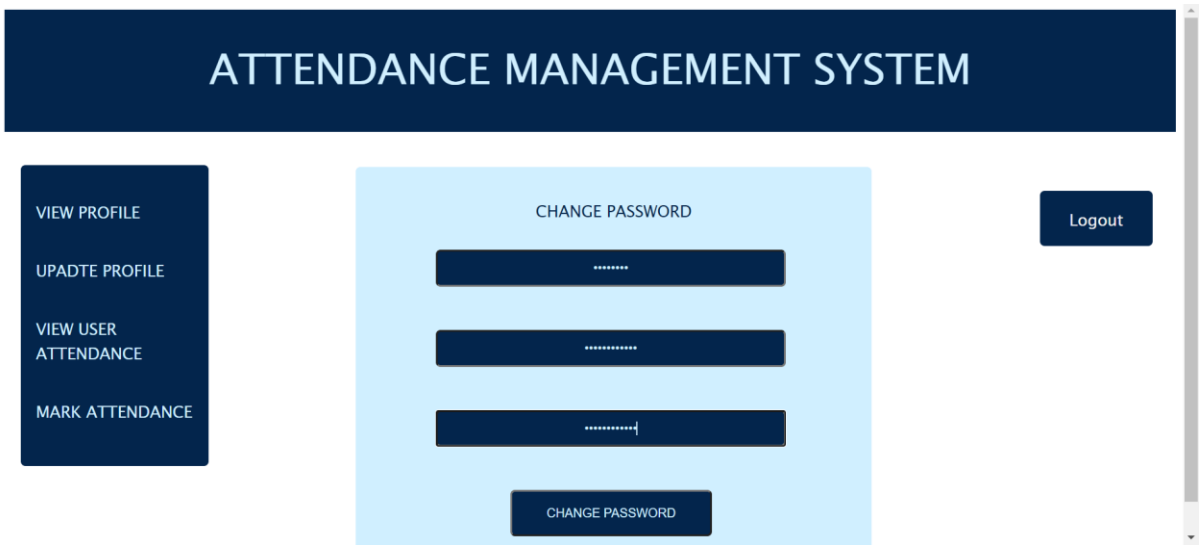
DATABASE BEFORE UPDATING PROFILE :

id	full_name	email_id	password	phone_no	date_of_birth
101	MATURI BHAVYA VENKATA NAGA SAI MANI	bhavyamaturi78592@gmail.com	A@Bhavya	6281919801	2001-10-23 20:20:53
102	DUDDU NIMSY	nimsiduddu@gmail.com	A@Nimsy	8520986627	2001-08-27 20:21:53
103	ADDANKI JAYA MADHURI	addankijayamadhuri11@gmail.com	A@Madhuri	6304600711	2001-06-10 20:25:37

DATABASE AFTER UPDATING PROFILE :

id	full_name	email_id	password	phone_no	date_of_birth
101	MATURI BHAVYA VENKATA NAGA SAI MANI	bhavyamaturi59826@gmail.com	A@Bhavya	8096528666	2001-10-23 20:20:53
102	DUDDU NIMSY	nimsiduddu@gmail.com	A@Nimsy	8520986627	2001-08-27 20:21:53
103	ADDANKI JAYA MADHURI	addankijayamadhuri11@gmail.com	A@Madhuri	6304600711	2001-06-10 20:25:37

ADMIN CHANGE PASSWORD :



DATABASE BEFORE CHANGING PASSWORD :

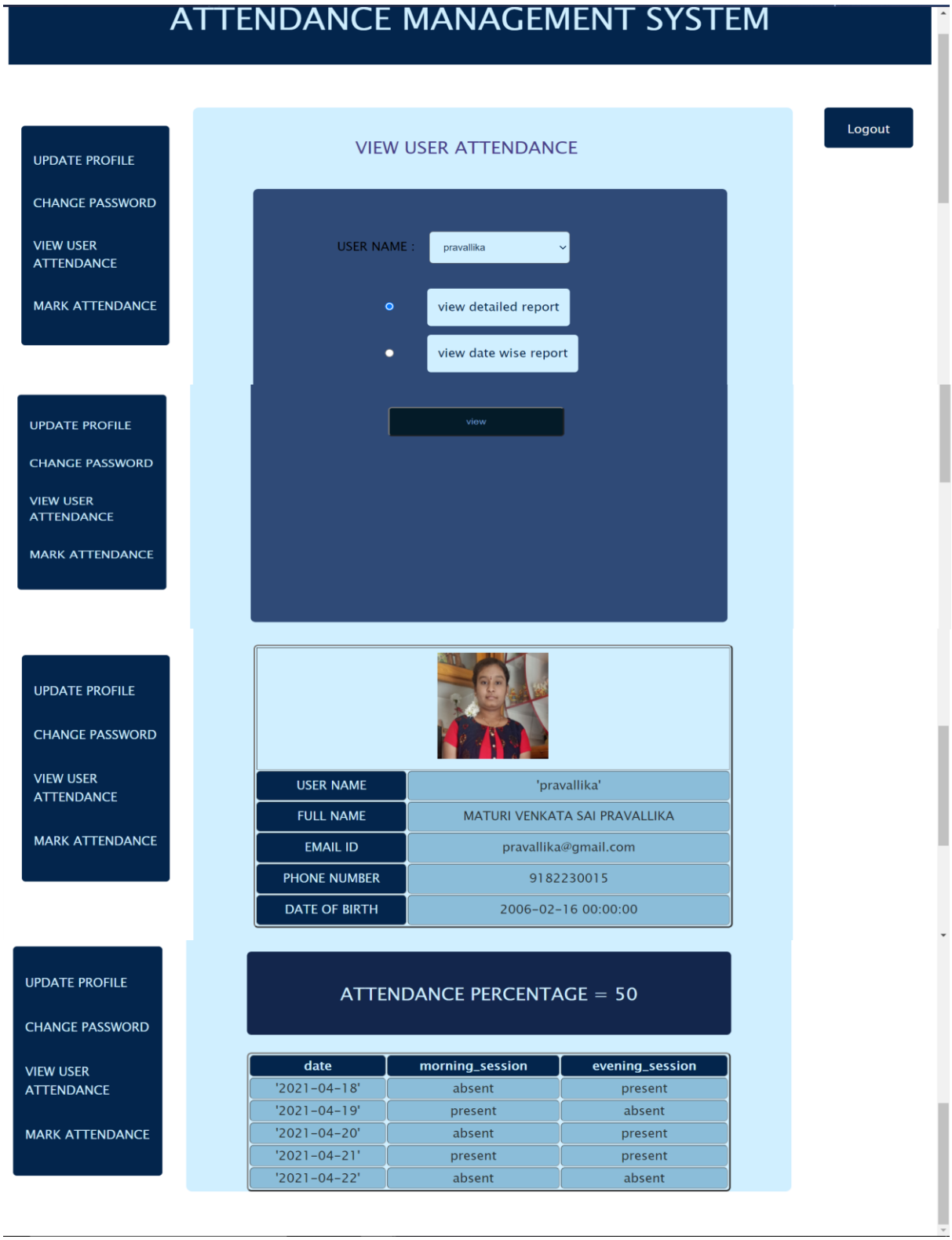
id	full_name	email_id	password	phone_no	date_of_birth
101	MATURI BHAVYA VENKATA NAGA SAI MANI	bhavyamaturi59826@gmail.com	A@Bhavya	8096528666	2001-10-23 20:20:53
102	DUDDU NIMSY	nimsiduddu@gmail.com	A@Nimsy	8520986627	2001-08-27 20:21:53
103	ADDANKI JAYA MADHURI	addankijayamadhuri11@gmail.com	A@Madhuri	6304600711	2001-06-10 20:25:37

DATABASE AFTER CHANGING PASSWORD :

id	full_name	email_id	password	phone_no	date_of_birth
101	MATURI BHAVYA VENKATA NAGA SAI MANI	bhavyamaturi59826@gmail.com	Admin@Bhavya	8096528666	2001-10-23 20:20:53
102	DUDDU NIMSY	nimsiduddu@gmail.com	A@Nimsy	8520986627	2001-08-27 20:21:53
103	ADDANKI JAYA MADHURI	addankijayamadhuri11@gmail.com	A@Madhuri	6304600711	2001-06-10 20:25:37

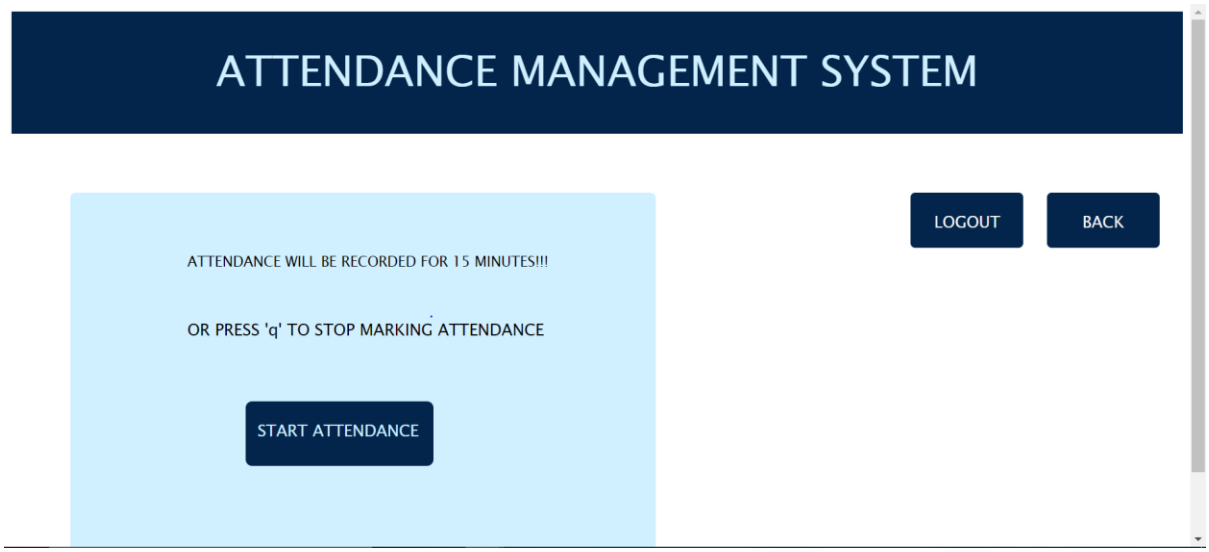
AFTER ADMIN CHANGING PASSWORD :

ADMIN VIEW USER ATTENDANCE :

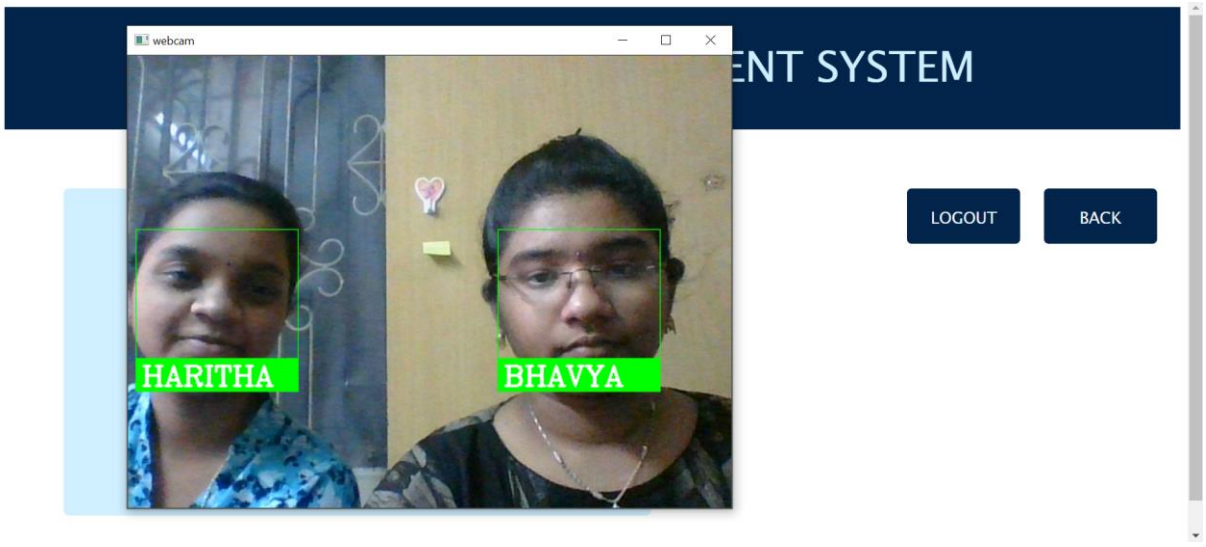


ADMIN VIEW DATEWISE REPORT :

ADMIN MARK ATTENDANCE :



START ATTENDANCE IN MORNING SESSION :



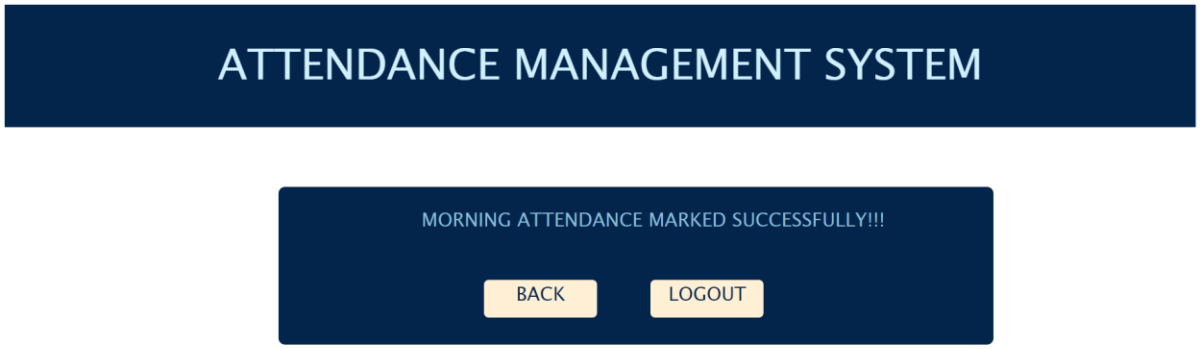
DATABASE BEFORE MORNING ATTENDANCE :

USER_ID	'2021-04-18'	'2021-04-19'	'2021-04-20'	'2021-04-21'	'2021-04-22'
43	5	3	0	1	0
44	5	5	3	3	0
48	5	5	3	3	0
52	5	5	5	3	0
53	3	1	3	5	0
54	5	3	1	5	0

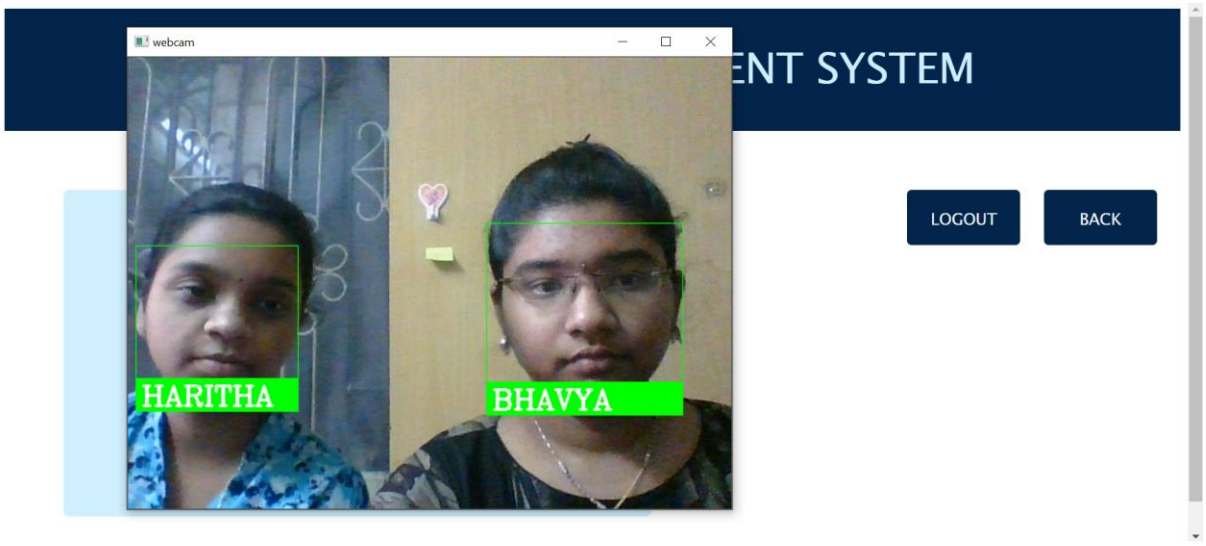
DATABASE AFTER MORNING ATTENDANCE :

USER_ID	'2021-04-18'	'2021-04-19'	'2021-04-20'	'2021-04-21'	'2021-04-22'
43	5	3	0	1	1
44	5	5	3	3	0
48	5	5	3	3	0
52	5	5	5	3	1
53	3	1	3	5	0
54	5	3	1	5	0

AFTER MORNING ATTENDANCE :



START ATTENDANCE IN AFTERNOON SESSION :



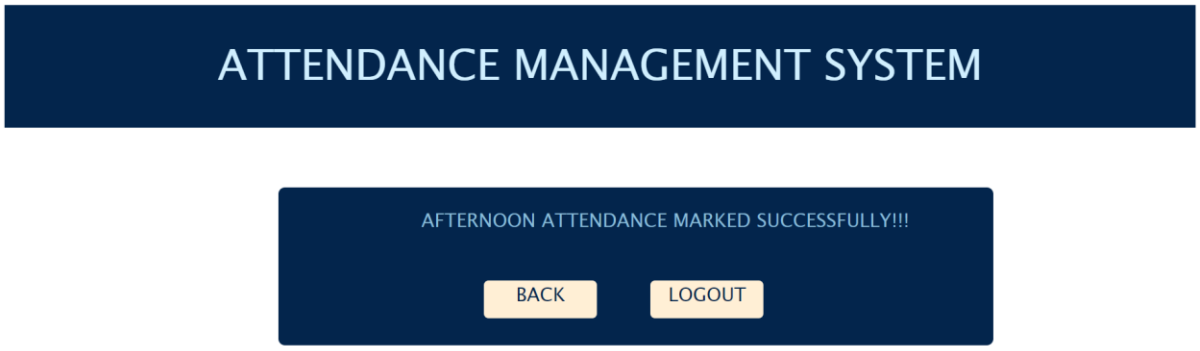
DATABASE BEFORE AFTERNOON ATTENDANCE :

USER_ID	'2021-04-18'	'2021-04-19'	'2021-04-20'	'2021-04-21'	'2021-04-22'
43	5	3	0	1	1
44	5	5	3	3	1
48	5	5	3	3	0
52	5	5	5	3	1
53	3	1	3	5	0
54	5	3	1	5	1

DATABASE AFTER AFTERNOON ATTENDANCE :

USER_ID	'2021-04-18'	'2021-04-19'	'2021-04-20'	'2021-04-21'	'2021-04-22'
43	5	3	0	1	3
44	5	5	3	3	1
48	5	5	3	3	0
52	5	5	5	3	3
53	3	1	3	5	0
54	5	3	1	5	1

AFTER AFTERNOON ATTENDANCE :



17. CONCLUSION

FACE RECOGNITION BASED ATTENDANCE MANAGEMENT SYSTEM is the system that facilitates for automatic maintainance of attendance for users using face recognition technique.This system saves time and energy for both users and admins.

The proposed system can be developed in :

- RECOGNITION OF FACE can be made more efficient in darkness.
- The proposed system can display time at which attendance is marked instead of present or absent.

REFERENCES

- Harvey M. Deitel and Paul J.Deitel, "Internet & World Wide Web How to Program", 4/e, Pearson Education.
- Jason Cranford Teague "Visual Quick Start Guide CSS, DHTML & AJAX", 4/ e, "Pearson Education".
- Roger S. Pressman, Software Engineering - A Practitioner's Approach, Seventh Edition, McGraw Hill Publications.
- Software Engineering Resources : - www.rspa.com/spi/
- Database Systems, Ramez Elmasri and Shamkant B.Navathe, Pearson Education, 6th edition.
- James Rumbaugh, Jacobson, Booch, Unified Modeling Language Reference Manual, 2nd Edition, PHI.

- https://www.eff.org/pages/face-recognition
- https://greenteapress.com/wp/learning-with-python/