



**Dharmsinh Desai University, Nadiad**  
**Faculty of Technology**  
**Department of Computer Engineering**

**B. Tech – CE, Semester: V**

**Subject:** (CE – 515) Advanced Technologies

**Project title:** Attendance Management System.

**Prepared by:** Birva Babaria (CE010, 19CEUON064)

Dharmi Chavda (CE024, 19CEUBS127)

**Guided by:** Prof. Prashant Jadav

Prof. Siddharth Shah

# **Contents**

1) Abstract .....	03
2) Introduction... ..	04
3) Software Requirements Specifications .....	05
4) Design	
a) Data flow diagram .....	09
b) ER diagram.....	10
c) Data dictionary .....	11
5) Implementation Details	
a) Modules .....	12
b) Major Functionality .....	13
6) Testing .....	16
7) Screen-shots.....	18
8) Conclusion.....	22
9) Limitation and future extension .....	23
10) Bibliography .....	23

# 1) **Abstract**

Attendance management system is the software developed for daily student or employee attendance in schools, universities, MNCs and various other institutions. In school and colleges attendance of students and faculties are marked whereas in companies and organizations attendance of employees is marked.

The attendance information is stored by the operators and accordingly grades would be allocated to students as per their attendance. Depending on employee attendance, salary will be decided by organization. So the management of the attendance is utmost important nowadays. It should be well recorded and properly managed and whenever any report history is required like monthly or yearly then it should be easily available to the operators who records attendance.

Nowadays using various technologies we can build some systems which reduce the work of the admin department where they are required to manage the attendance using register and make report from it, that seems to be boring and tedious task. Such software application can save a lot of time and ultimately can be more convenient.

## 2) Introduction

Attendance management system is basically a software that records the presence of the employee. The purpose of developing this software is to computerize the traditional way of taking attendance.

Firstly, the registered user can successfully login to the system. This system is specifically developed for the offices where operators can enter the records of the working employee and mark their attendance. Information entered can further be edited or deleted permanently. All the records would be stored according to date and would be visible in attendance history.

### Technologies/Tools used

#### **Technologies:**

- React JS
- Node JS
- MongoDB
- Express
- CSS
- Bootstrap

#### **Tools:**

- Git
- Visual studio code

### **3) Software Requirement Specifications**

#### **1) Manage operators**

##### **R.1.1 : Profile**

***Description:*** This section includes information related to operator's profile.

##### **R.1.1.1 : View profile**

***Description:*** This option shows the profile of the operator. It displays their username and email id.

***Input:*** Click on the view profile option.

***Output:*** Profile screen would be displayed.

##### **R.1.1.2 : Edit profile**

***Description:*** This option edits the profile of the Operator.

***Input:*** Click on the edit profile button .

***Output:*** Profile would be updated.

##### **R.1.2 : Contact us page**

***Description:*** This function will let operator to see the information about website and can contact admin through mail or contact number.

***Input:*** Click on the contact us option from the side bar.

***Output:*** Contact us page would be displayed.

##### **R.1.3 : About us page**

***Description:*** This function will let operator to see the information related to admin.

***Input:*** Click on the about us option from the side bar.

***Output:*** About us page would be displayed.

## **2) Manage registration/login**

### **R.2.1 : Operator**

#### **R.2.1.1 : Registration**

***Description:*** If Operator doesn't have any exiting account then they have to register themselves.

***Input:*** User have to provide their name, email, and password.

***Output:*** User would be redirected to login page.

#### **R.2.1.2 : Login**

***Description:*** If Operator already have an account then this option will be used to display home page by logging in.

***Input:*** Operator have to give their username and password.

## **3) Manage Employees**

### **R.3.1 : Add record**

***Description:*** This function will allow operator to add the employee information along with its status.

***Input:*** Click on the Add Record.

***Output:*** Add record form would be displayed.

### **R.3.2 : View employees record**

***Description:*** This function will show the list of all the employee records along with date.

***Input:*** Click on the Modify record.

**Output:** List of all the entered employees would be visible.

### **R.3.3 : Edit record**

**Description:** This function will let operator to edit any employee record and save changes to the database.

**Input:** Click on buy edit record button.

**Output:** edit form would be visible where operator can edit and save changes to database.

### **R.3.4 : Delete record**

**Description:** This function will let operator to delete any employee record.

**Input:** Click on delete record button.

**Output:** Employee record would be deleted from the list.

### **R.3.5 : View record history**

**Description:** This function will let operator to view history of all the entered employees.

**Input:** Click on Attendance history button from sidebar.

**Output:** Employees records history would be visible.

### **R.3.6 : Search employees**

**Description:** This function will let operator search all the recorded employees with respect to name, category, date and email id.

**Input:** Type text in the search bar.

**Output:** Matched employee records would be visible accordingly.

#### **4) Logout user**

##### **R.7.1 : logout**

**Description:** Operator can logout after completion of task. All the entered records would not be affected due to logout.

**Input:** Click on logout option in side bar.

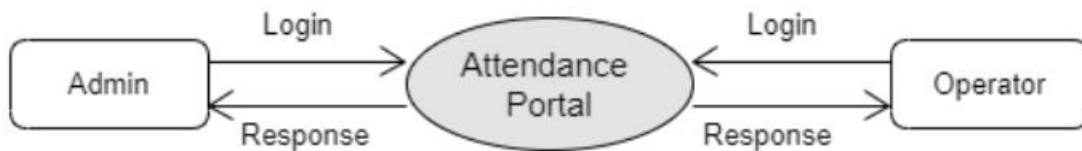
**Output:** Operator would be redirected to login page again.



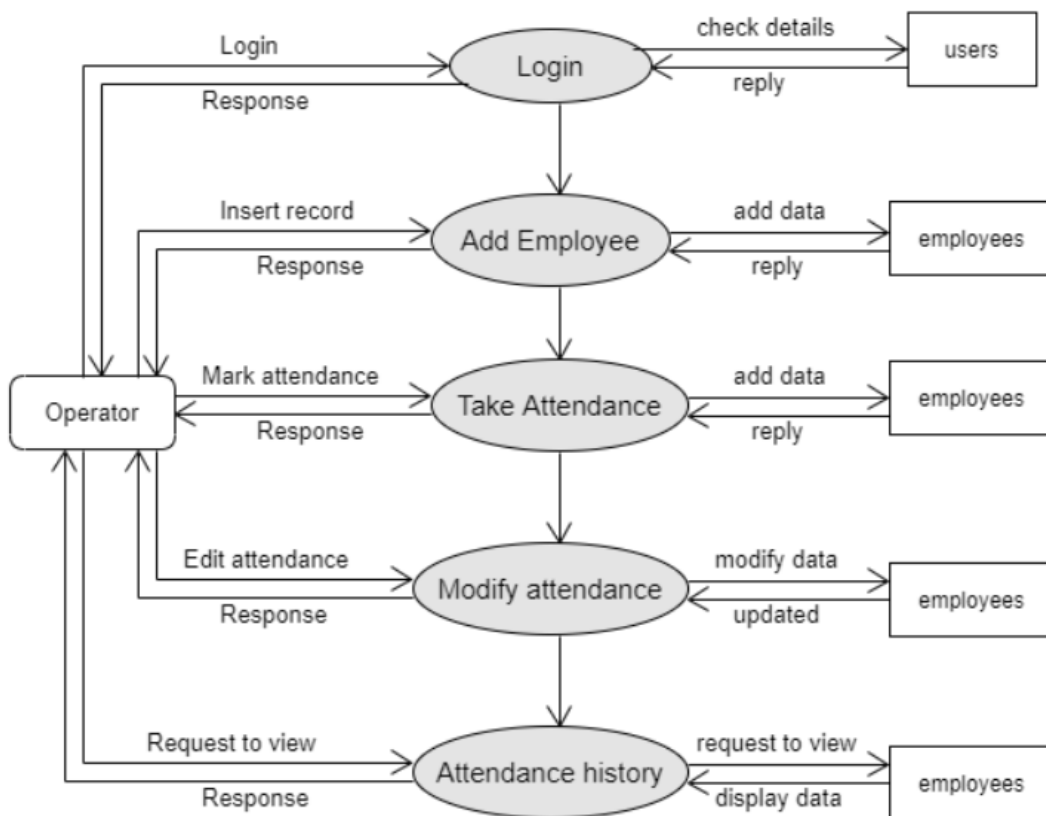
## 4) Design

a) DFD diagram

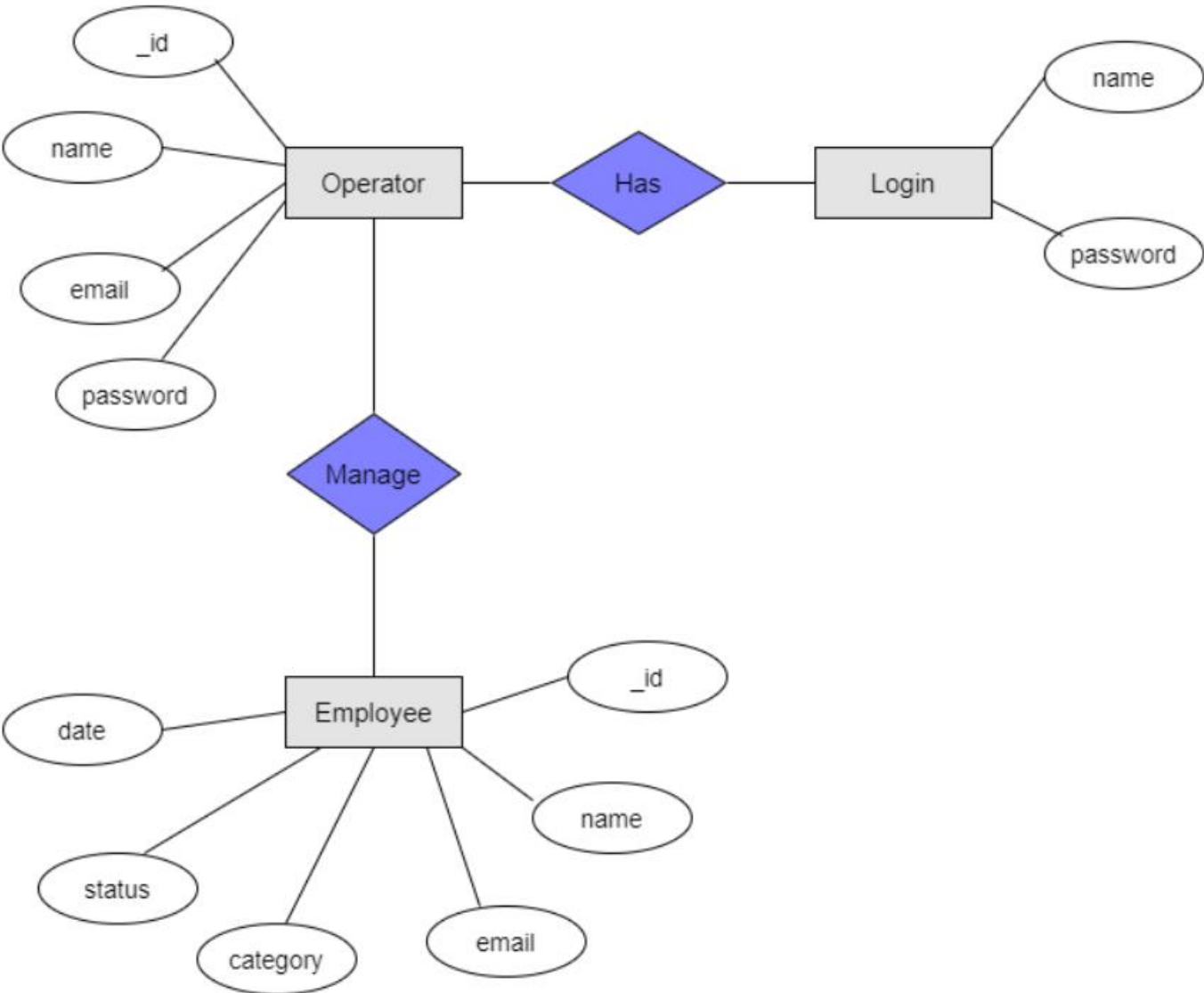
### Zero Level DFD (Context level)



### Level 1 DFD



b) E-R diagram



c) Data dictionary

<b>Users</b>							
Sr No.	Field name	Data type	Required	Unique	PK/FK	Referred table	Description
1	_id	Int32	yes	yes	yes	-	Auto increment
2	name	String	yes	yes	no	-	-
3	email	String	yes	no	no	-	-
4	password	String	yes	no	no	-	-

<b>Employees</b>							
Sr No.	Field name	Data type	Required	Unique	PK/FK	Referred table	Description
1	_id	Int32	yes	yes	yes	-	Auto increment
2	name	String	yes	yes	no	-	-
3	email	String	yes	no	no	-	-
4	category	String	yes	no	no	-	-
5	status	String	yes	no	no	-	-

## **5) Implementation Details**

### **a). Modules**

Each module consists of several methods to implement the required Functionality. Implementation is done using Node JS. Database used in these modules is MongoDB.

#### **Login-Registration**

This module is the base for authentication and authorization to ensure the security aspect of the user.

It consist of all the login and registration functionality. User can login to the system if account already exist. User can register with unique username to create an account.

User is supposed to provide correct credentials to successfully login to the system.

User can logout from the system whenever he/she wants. This functions would not delete the activities performed by user.

#### **Employee Records**

This module is basically used in entering the record of the employees and mark their attendance. Once we enter the information and click on add record the following information would be added to the database.

Operator can edit the information of employee by clicking on the edit button. All the updated fields will be added to the database.

Operator can delete the record of any particular employee by clicking on the delete button. That particular column would be deleted from the list of the employee.

Operator can search the employee from the list with respect to either name, category, status or date.

## b). Major functionalities

- Login functionality

```
app.post("/login", (req, res) => {
  const {name, password} = req.body
  User.findOne({name: name}, (err, user) => {
    if(user){
      if(password === user.password){
        res.send({message: "Login successful", user: user})
      }
      else{
        res.send({message: "Password didn't match"})
      }
    }
    else{
      res.send({message: "User not registered"})
    }
  })
})
```

- Register functionality

```
app.post("/register", (req, res) => {
  const {name, email, password} = req.body
  User.findOne({name: name}, (err, user) => {
    if(user){
      res.send({message: "User already registered"})
    }
    else{
      const user = new User({
        name,
        email,
        password
      })
      user.save(err => {
        if(err) {
          res.send(err)
        }
        else{
          res.send({message: "Successfully Registered, please login now!"})
        }
      })
    }
  })
})
```

- Get all users

```
// Get all users
export const getUsers = async (request, response) => {
  try{
    const users = await User.find();
    response.status(200).json(users);
  }catch( error ){
    response.status(404).json({ message: error.message })
  }
}
```

- Get user by ID

```
// Get a user by id
export const getUserById = async (request, response) => {
  try{
    const user = await User.findById(request.params.id);
    response.status(200).json(user);
  }catch( error ){
    response.status(404).json({ message: error.message })
  }
}
```

- Add user

```
// Save data of the user in database
export const addUser = async (request, response) => {
  // retrieve the info of user from frontend
  const user = request.body;
  console.log("inside")

  const newUser = new User(user);
  try{
    await newUser.save();
    response.status(201).json(newUser);
  } catch (error){
    response.status(409).json({ message: error.message});
  }
}
```

- Edit user

```
// Save data of edited user in the database
export const editUser = async (request, response) => {
  let user = await User.findById(request.params.id);
  user = request.body;

  const editUser = new User(user);
  try{
    await User.updateOne({_id: request.params.id}, editUser);
    response.status(201).json(editUser);
  } catch (error){
    response.status(409).json({ message: error.message});
  }
}
```

- Delete user

```
// deleting data of user from the database
export const deleteUser = async (request, response) => {
  try{
    await User.deleteOne({_id: request.params.id});
    response.status(201).json("User deleted Successfully");
  } catch (error){
    response.status(409).json({ message: error.message});
  }
}
```

## 6) Testing

**Testing method:** Manual testing

Manual testing was performed in order to find and fix the bugs in development process.

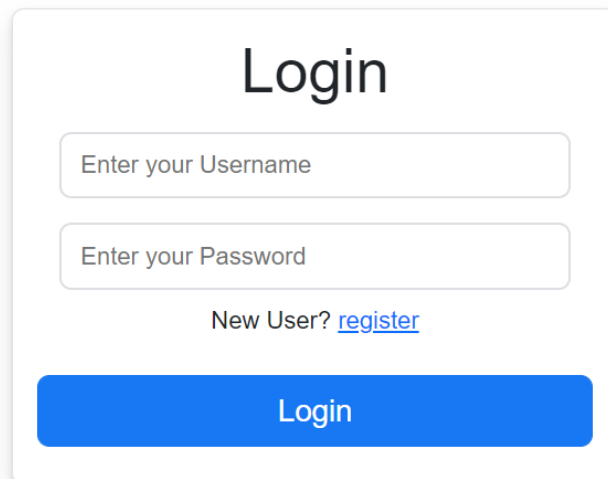
Sr no.	Test Scenario	Expected result	Actual result	Status
1	Enter to the website	Login Page should be displayed	Login page is displayed	Success
2	Login with correct credentials	User should be able to login	User is redirected to home page	Success
3	Login with incorrect credentials	Alert box is popped up with a message	Alert box is popped up	Success
4	Registering with registered username	'user already registered' alert box should be displayed	Alert box is popped up	Success
5	Entering different password and confirm password	'password does not match' alert box should be displayed	'password does not match' alert box is displayed	Success
6	Clicking on login/register button with empty fields	'invalid input' alert box should be appeared	'invalid input' alert box is appeared	Success
7	Logout	User should logged out from the system	User is logged out from the system and cannot use the system until next login	Success
8	Add record to the database	Record should be added to the list of all the employees	Record is added to the list	Success



9	Click on modify information	List of all employees should be appeared	List of all employees is displayed	Success
10	Edit record	Edit form should be visible	Edit form is visible	Success
11	Click on edit user	Record should be updated	Record is updated	Success
12	Click on delete user	Record should be deleted	Record is deleted	Success
13	Click on attendance history	List of all recorded employees should be visible	List of all recorded employees is visible	Success
14	Search employees	Matched employees list should be visible	Matched employees list is visible	Success
15	Click on Contact us page	Contact us page should be visible	Contact us page is visible	Success
16	Click on About us	About us page should be visible	About us page is visible	Success
17	Click on Profile option	Profile should be visible	Profile is visible	Success
18	Click on edit profile button	Profile should be updated	Profile is updated	Success

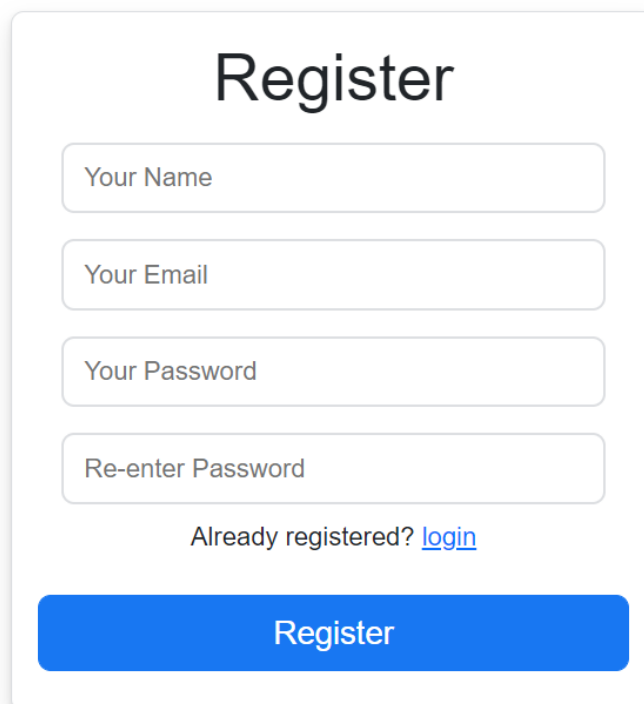
## 7) Screen-shots

- Login page



A screenshot of a login form. The form is titled "Login" in a large, bold, black font. Below the title, there are two input fields: "Enter your Username" and "Enter your Password". Below these fields, there is a link "New User? [register](#)". At the bottom of the form, there is a blue button labeled "Login".

- Register page



A screenshot of a register form. The form is titled "Register" in a large, bold, black font. Below the title, there are four input fields: "Your Name", "Your Email", "Your Password", and "Re-enter Password". Below these fields, there is a link "Already registered? [login](#)". At the bottom of the form, there is a blue button labeled "Register".

- Add record

The screenshot shows a web browser at localhost:3000 displaying the 'Attendance-Management' application. On the left is a dark sidebar with navigation links: 'Attendance History', 'Modify Information', 'Add Employee Record', 'About', and 'Contact us'. At the bottom of the sidebar is a user profile icon and the text 'ADMIN'. The main content area is titled 'ADD RECORD' and contains a form with five input fields: 'Name', 'Email', 'Category', 'Status', and 'Date'. Below these fields is a blue button labeled 'ADD RECORD'.

- Modify record

The screenshot shows the 'EMPLOYEE LIST' page in the 'Attendance-Management' application. The left sidebar is identical to the previous screenshot. The main content area displays a table with the following data:

Id	Name	Email	Category	Status	Date	Action
0	org1	b123@g.com	admin	P	12/01/2012	<div>EDIT</div> <div>DELETE</div>
1	dharmi	k123@g.com	HR	A	22/01/2010	<div>EDIT</div> <div>DELETE</div>
3	vinod	b123@g.com	HR	P	16/11/2021	<div>EDIT</div> <div>DELETE</div>
6	kailas	o123@g.com	admin	P	12/01/2007	<div>EDIT</div> <div>DELETE</div>
						<div>EDIT</div>

- Attendance history

Attendance-Management

Attendance History

Modify Information

Add Employee Record

About

Contact us

ADMIN

ALL HISTORY

Search here...

_id	name	email	category	status	__v	date
0	org1	b123@g.com	admin	P	0	12/01/2012
1	dharmi	k123@g.com	HR	A	0	22/01/2010
3	vinod	b123@g.com	HR	P	0	16/11/2021
6	kailas	o123@g.com	admin	P	0	12/01/2007
7	abc	o123@g.com	HR	P	0	12/01/2001

- Profile

Attendance-Management

Attendance History

Modify Information

Add Employee Record

About

Contact us

ADMIN

EDIT PROFILE

Name

ADMIN

Email

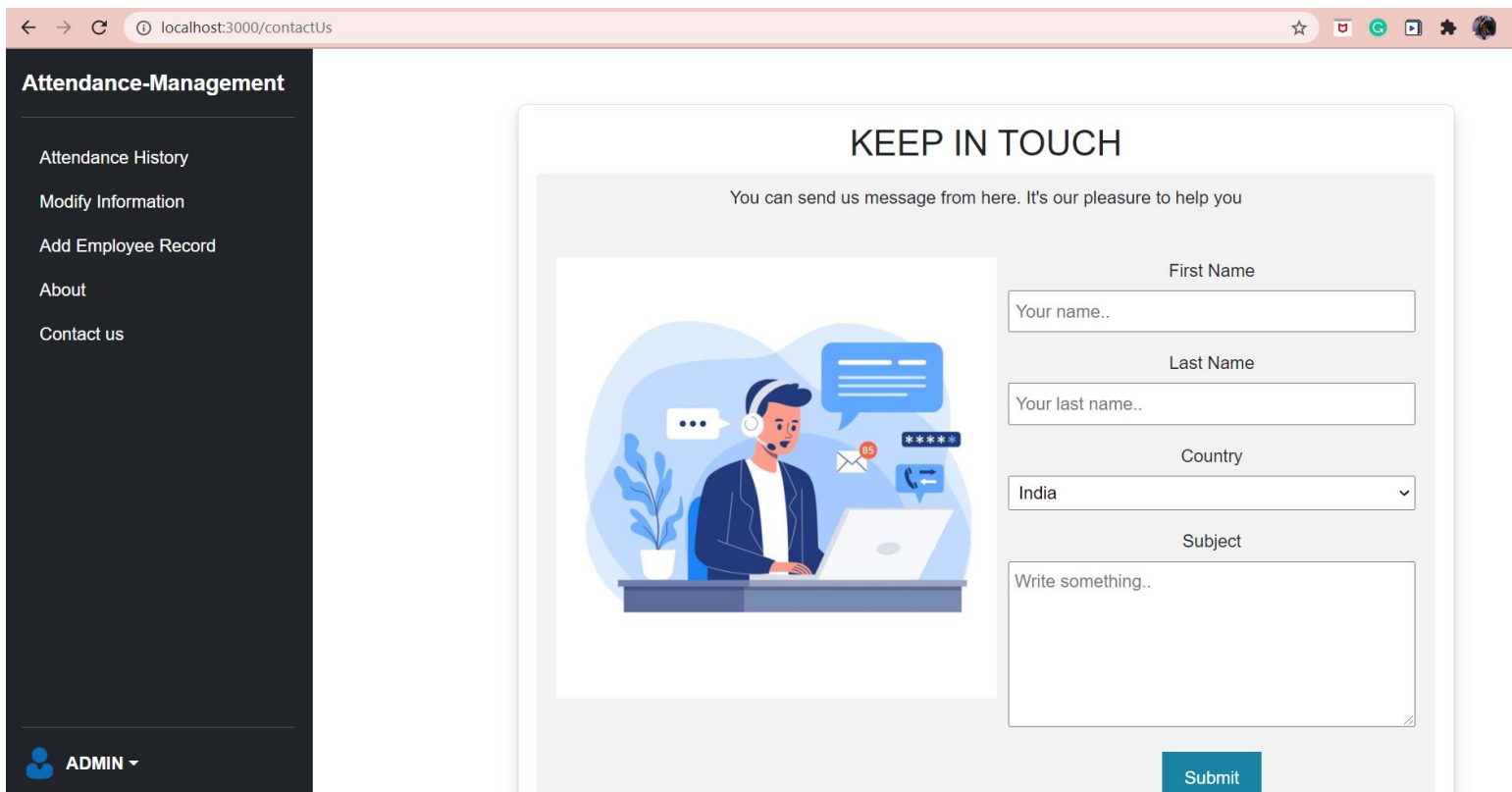
admin@gmail.com

Category

ADMIN

EDIT PROFILE

- Contact us



Attendance-Management

- Attendance History
- Modify Information
- Add Employee Record
- About
- Contact us

ADMIN ▾

## KEEP IN TOUCH

You can send us message from here. It's our pleasure to help you

First Name  
Your name..

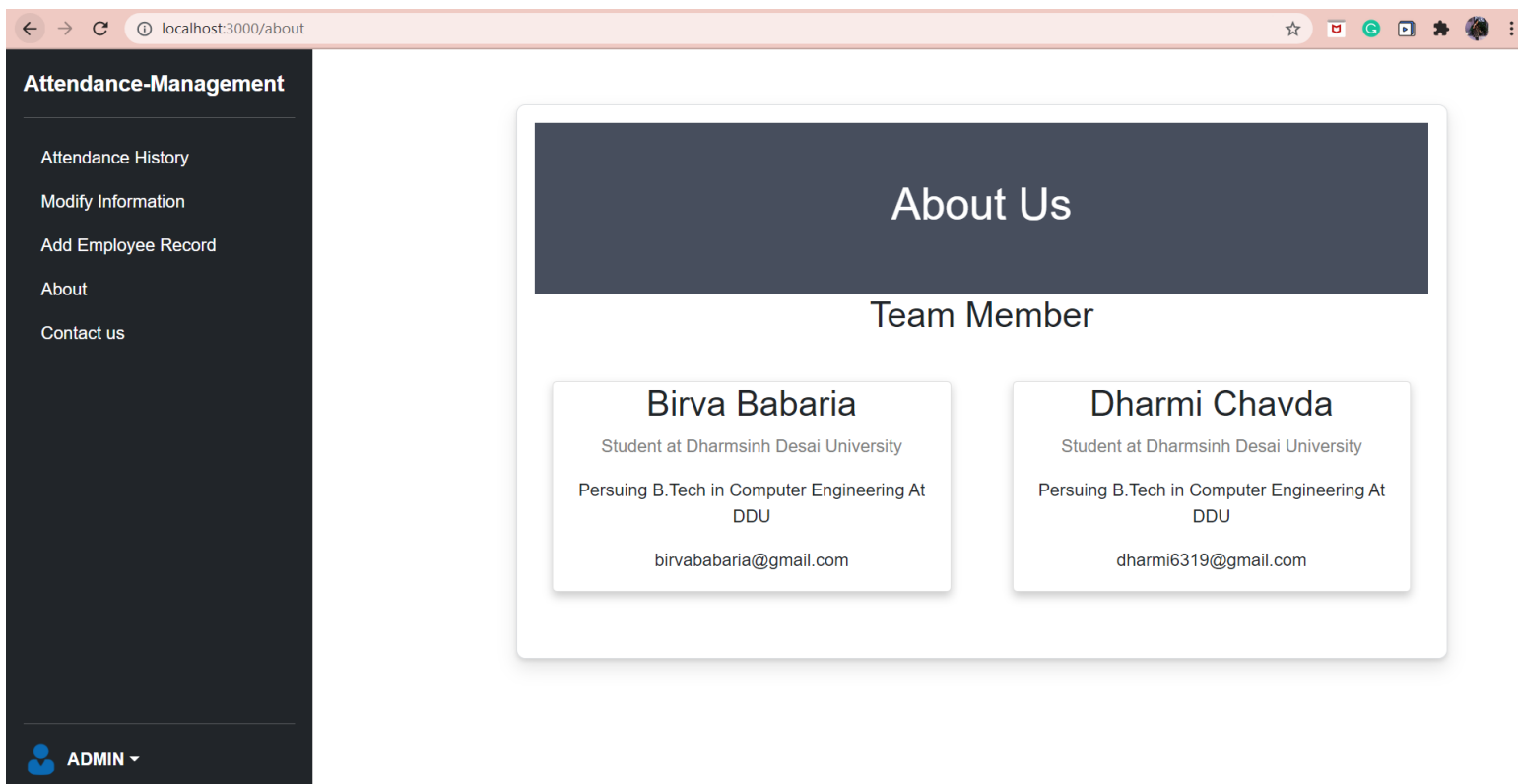
Last Name  
Your last name..

Country  
India ▾

Subject  
Write something..

Submit

- About us



Attendance-Management

- Attendance History
- Modify Information
- Add Employee Record
- About
- Contact us

ADMIN ▾

## About Us

### Team Member

**Birva Babaria**  
Student at Dharmsinh Desai University  
Persuing B.Tech in Computer Engineering At DDU  
birvababaria@gmail.com

**Dharmi Chavda**  
Student at Dharmsinh Desai University  
Persuing B.Tech in Computer Engineering At DDU  
dharmi6319@gmail.com

## 8) **Conclusion**

The functionalities that are implemented in the system are prepared after understanding all functionalities according to software requirements specifications (SRS). Functionalities that are successfully implemented in the system are as follows:

- Login
- Registration
- User authentication
- Logout
- Add employee record
- Add attendance
- Edit employee record
- Delete employee record
- Search employee record
- View attendance history
- Edit profile

After implementing all these functionalities, comprehensive testing was performed on the system to determine possible errors.

## 9) **Limitations and future extensions**

### **Limitations:**

- Employees will not be able to interact directly with the system.
- This project is suitable for small scale organization.
- Operator would not be able to record the employee's attendance according to its category.

### **Future extensions:**

- Creating employee interface.
- Records can be stored in more organized manner to make it more convenient.

## 10) **Bibliography**

Following links and websites were referred during the development of this project:

- <https://docs.mongodb.com/manual/>
- <https://github.com/>
- <https://stackoverflow.com/>
- <https://reactjs.org/>
- <https://www.w3schools.com/nodejs/>