SUMMER INTERNSHIP REPORT

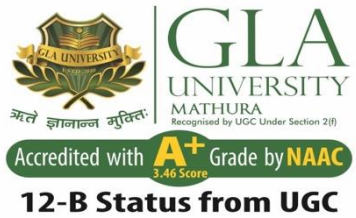
On

CRUD Application

Submitted by

Radhika Tiwari
2115000795

Department of Computer Engineering & Applications
Institute of Engineering & Technology



GLA University
Mathura- 281406, INDIA
2023

Department of computer Engineering and Applications
GLA University, Mathura
17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,
Mathura – 281406

Declaration

I hereby declare that the work which is being presented in the Summer Internship **"CRUD Application"** in partial fulfillment of the requirements for Summer Internship viva voce, is an authentic record of my own work carried under the supervision of Ms. Shreya Tandon at AppSquadz Software Pvt.Ltd., Noida, Uttar Pradesh during 22nd May 2023 to 12th July 2023.

(Signature of Candidate)

Name of Candidate: Radhika Tiwari

Roll. No.: 2115000795

Course: B.tech(CS)

Semester & Year: 5th &(3rd Year)

Section: E

Contents

Abstract	4
Certificate	5
Acknowledgments	6
1.Introduction	**7-8**
1.1 Motivation and Overview	7
1.2 Objectives	8
2.Software Requirement Analysis	**9**
2.1 Define the problem	9
2.2 Define the modules and their functionalities (SRS)	9
3.Software Design	**10-14**
3.1 Data Flow Diagram	10-12
3.2 UML Diagram	13
3.3 E-R Diagram	14
4.Testing	**15-16**
4.1 Black Box testing	15
4.2 White Box Testing	16
5.Implementation and User Interface	**17-20**
5.1 HTML Structure	17
5.2 CSS Styling	18
5.3 JavaScript Functionalities	19
5.4 Local Storage	20
References	**21**
6.Appendices	**22-34**

Abstract

CRUD is a type of mechanism that allows you to create data, read data, edit it, and delete those data. In my case, I am going to make a CRUD application, so we will have 4 options to create tasks, read tasks, update tasks, or delete tasks.

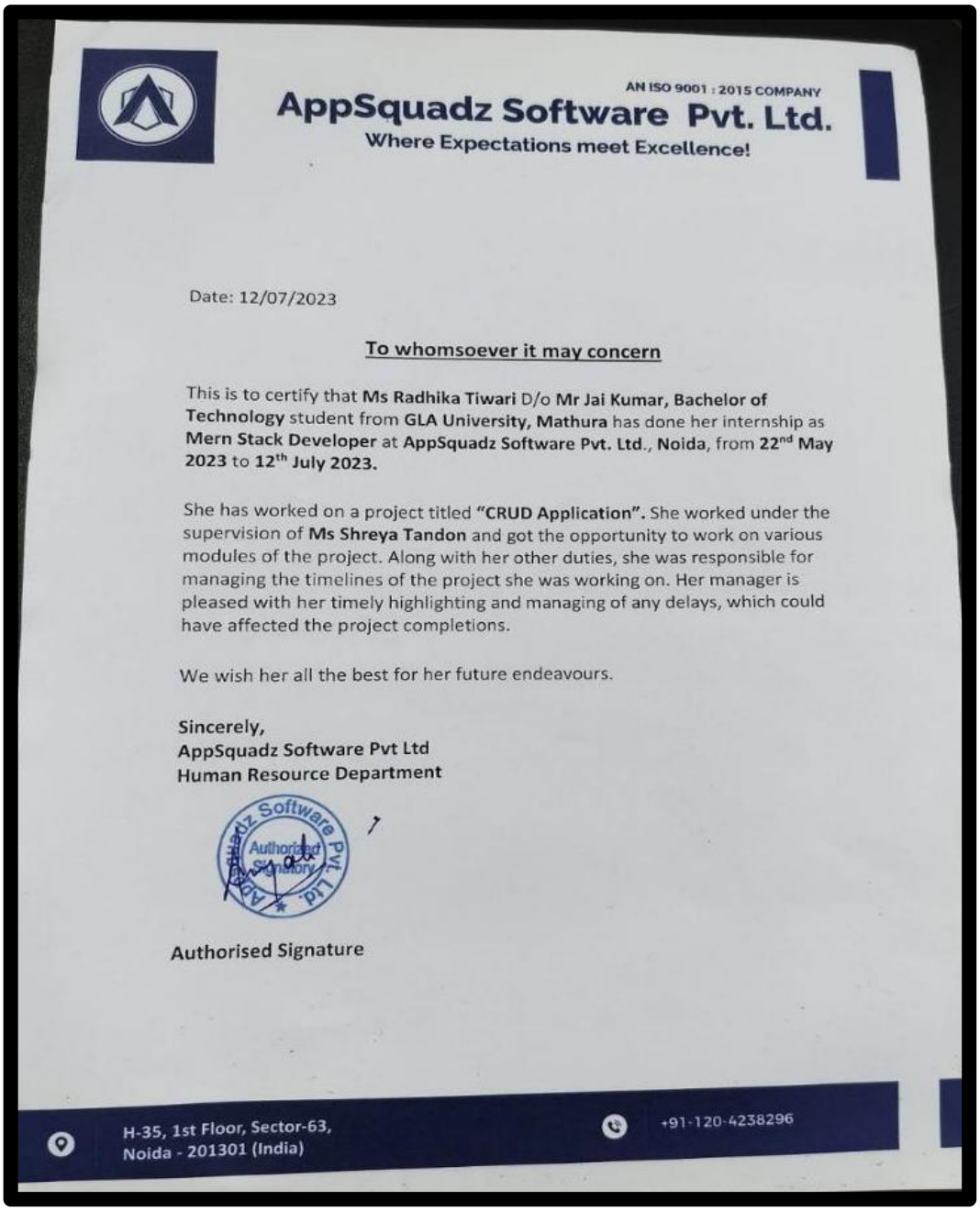
For this project, we will be following these steps below:

- Create 3 files named index.html, style.css, and main.js
- Link the JavaScript and CSS file to index.html
- Start your live server

The scope of this project includes:

- Designing a user-friendly interface using HTML and CSS.
- Implementing CRUD functionality using JavaScript.
- Allowing users to create, read, update, and delete items in the application.
- Ensuring data persistence using local storage.

Certificate



AN ISO 9001 : 2015 COMPANY

AppSquadz Software Pvt. Ltd.

Where Expectations meet Excellence!

Date: 12/07/2023

To whomsoever it may concern

This is to certify that **Ms Radhika Tiwari** D/o **Mr Jai Kumar, Bachelor of Technology** student from **GLA University, Mathura** has done her internship as **Mern Stack Developer** at **AppSquadz Software Pvt. Ltd.**, **Noida**, from **22nd May 2023** to **12th July 2023.**

She has worked on a project titled **"CRUD Application".** She worked under the supervision of **Ms Shreya Tandon** and got the opportunity to work on various modules of the project. Along with her other duties, she was responsible for managing the timelines of the project she was working on. Her manager is pleased with her timely highlighting and managing of any delays, which could have affected the project completions.

We wish her all the best for her future endeavours.

Sincerely,
AppSquadz Software Pvt Ltd
Human Resource Department

AppSquadz Software Pvt. Ltd. Authorised Signatory

Authorised Signature

H-35, 1st Floor, Sector-63,
Noida - 201301 (India)

+91-120-4238296

Acknowledgement

I am sincerely grateful to all the individuals who have supported and guided me during the course of my internship project on CRUD Application. This project has been a significant learning experience.

I would like to express my heartfelt gratitude to my internship supervisor, **Ms. Shreya Tandon**, for their constant guidance, expertise, and encouragement. Their mentorship has been instrumental in shaping the direction of this project and expanding my understanding of hospital operations.

I extend my thanks to the entire team at AppSquadz for providing me with the opportunity to work on this project within their esteemed institution.

I extend my gratitude to my friends and family for their unwavering encouragement and understanding during the ups and downs of this internship. Your belief in my abilities has been a constant motivation.

In conclusion, this internship project has been a remarkable experience, and I am deeply grateful to all those who have been a part of this journey. Each interaction and learning opportunity has contributed to my personal and professional growth.

1. Introduction

1.1 Motivation and Overview:

The motivation behind this project was to create a hands-on learning experience in web development. The project aims to provide a practical understanding of CRUD operations and their implementation using HTML, CSS, and JavaScript. The application's scope includes a user-friendly interface for managing data.

1.2 Objectives:

The primary objectives of this project are as follows:

- To design and develop a user interface for CRUD operations.
- To implement Create, Read, Update, and Delete functionality using JavaScript.
- To ensure data persistence using local storage.
- To enhance skills in web development and user interface design.

1.3 Project Structure:

- The project consists of the following files and directories.
- index.html: The main HTML file that contains the structure of the application.
- style.css: The CSS file responsible for styling the application.
- script.js: The JavaScript file containing the CRUD operations and event handling.
- images/: Directory containing any images used in the project.

2. Software Requirement Analysis

2.1 Define the Problem:

The problem addressed by this project is the need for a simple yet functional CRUD application that can be used to manage a list of items. The application should allow users to perform basic data operations without the need for complex interactions.

2.2 Define the Modules and their Functionalities (SRS):

The Software Requirements Specification (SRS) outlines the following modules and their functionalities:

User Interface: Provides an interactive interface for users to interact with the application.

Create Module: Allows users to add new items to the list.

Read Module: Displays the list of items to users.

Update Module: Enables users to edit and update existing items.

Delete Module: Allows users to remove items from the list.

Local Storage: Ensures data persistence by storing data locally on the user's device.

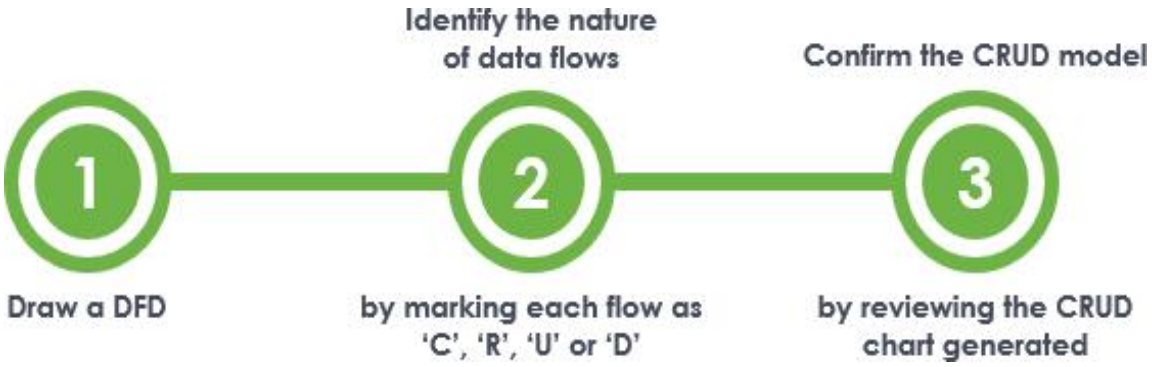
3. Software Design

3.1 Data Flow Diagram:

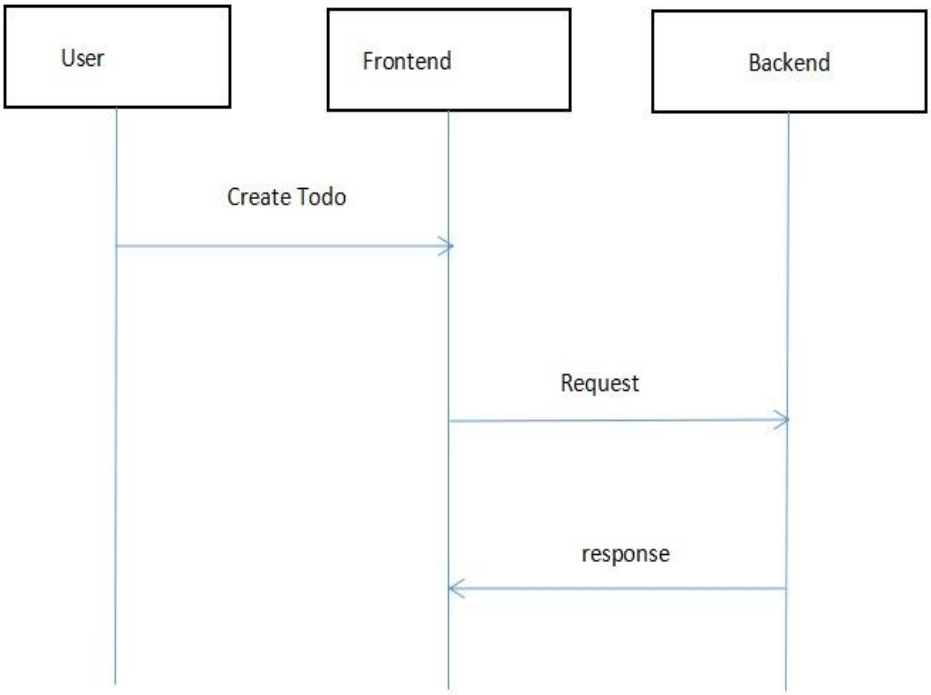
A Data Flow Diagram (DFD) is a graphical representation that shows how data flows through a system. In the context of a CRUD application, the DFD can illustrate the flow of data between various components such as users, the application's frontend, backend, and the database.

Objectives:

- Interrelate the relationship between a single process to a data store with CRUD matrix of database operations, (CREATE, READ, UPDATE, DELETE)
- Visualize what CRUD operations are required by the processes for acting on the data store in a matrix format.

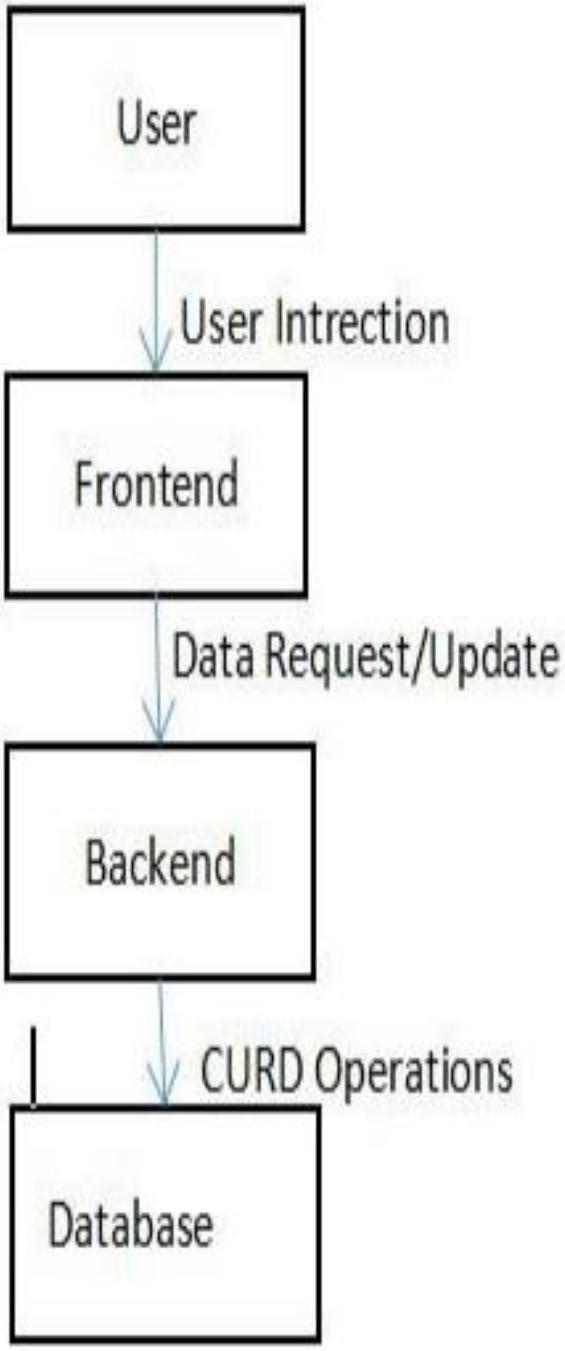


The figure above outlines how a CRUD Chart can be created from a DFD. In this article we will walk you through the steps in detail.



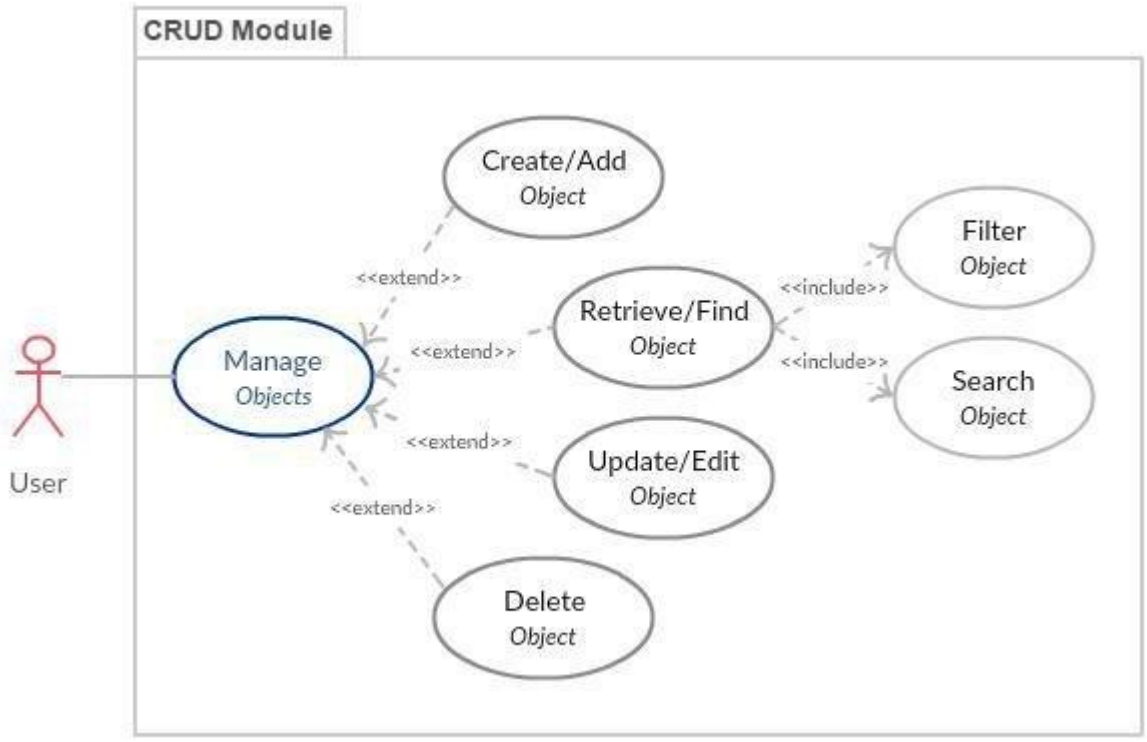
Indicate the nature of data flows, which are classified as:

- C - The data flow creates data in the data store connected
- R - The data flow receives data from the connected data store and forward it to another object on the diagram
- U - The data flow receives data from the connected source and updates the data in data store
- D - The data flow deletes data in data store.


User
User Intrection
Frontend
Data Request/Update
Backend
CURD Operations
Database

3.2 UML Diagram:

CRUD Module is the basic functionality offered in the system.
Manage objects is the base use case in CRUD Module where object is a term that could represent an entity in the system, such as User,
Role, Branch, etc.
Then, it would be divided into four basic use cases in accordance with its specific function of CRUD along with specified permission for each of them.

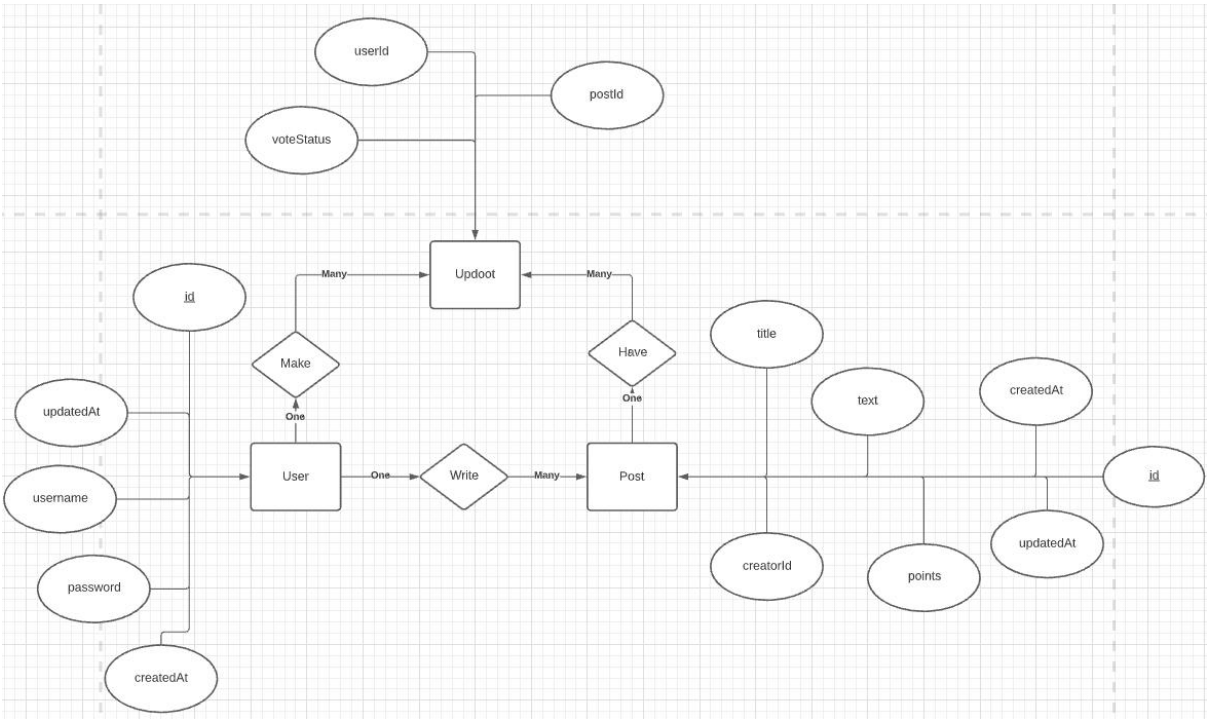


3.3 E-R Diagram:

A CRUD diagram shows what types of processing is performed on data by a system, indicating them in a matrix format for each function. The types are described as "Create," "Read," "Update" and "Delete," which are the operation types when data is manipulated by a database management system (DBMS). The diagram is called CRUD after the initials of these types.

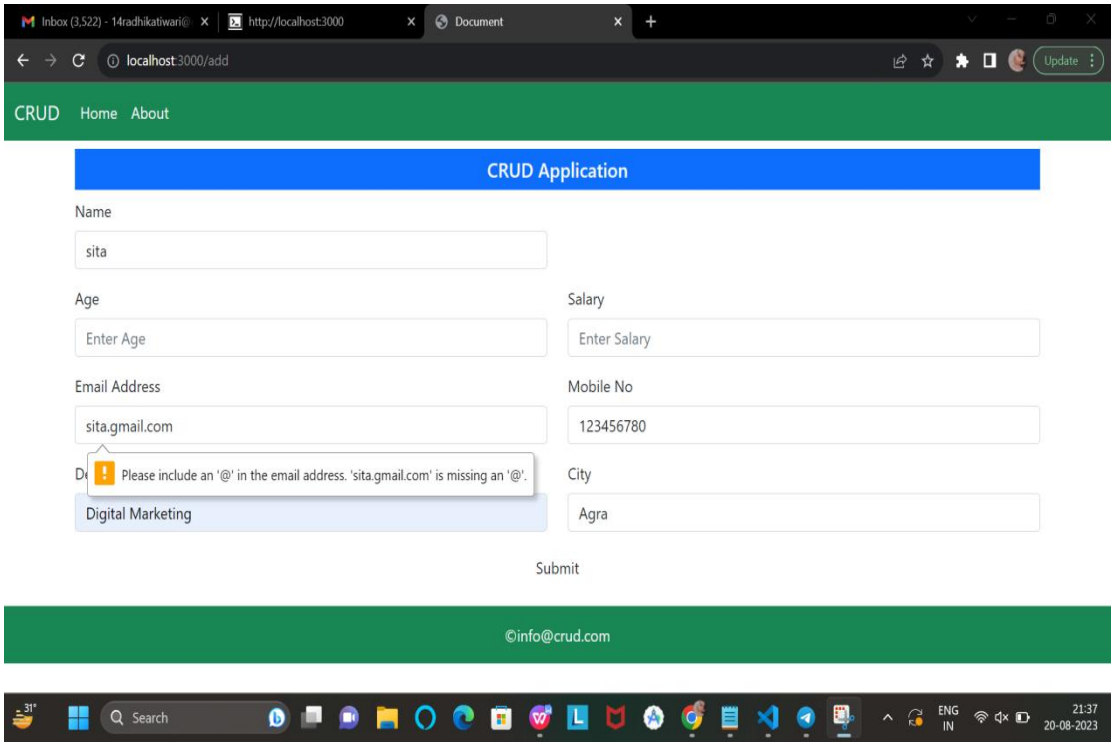
When the "function to change product name" is employed to change the data recorded as "product name" in the database, for example, a CRUD diagram that places functions in columns and data in rows shows "Update" at the intersection of the "function to change product name" (column) and "product name" (row).

When used in the design process of a system, a CRUD diagram helps verify if there is any lack of certain functions, omission of investigations on the database and concentration of processing tasks, etc.

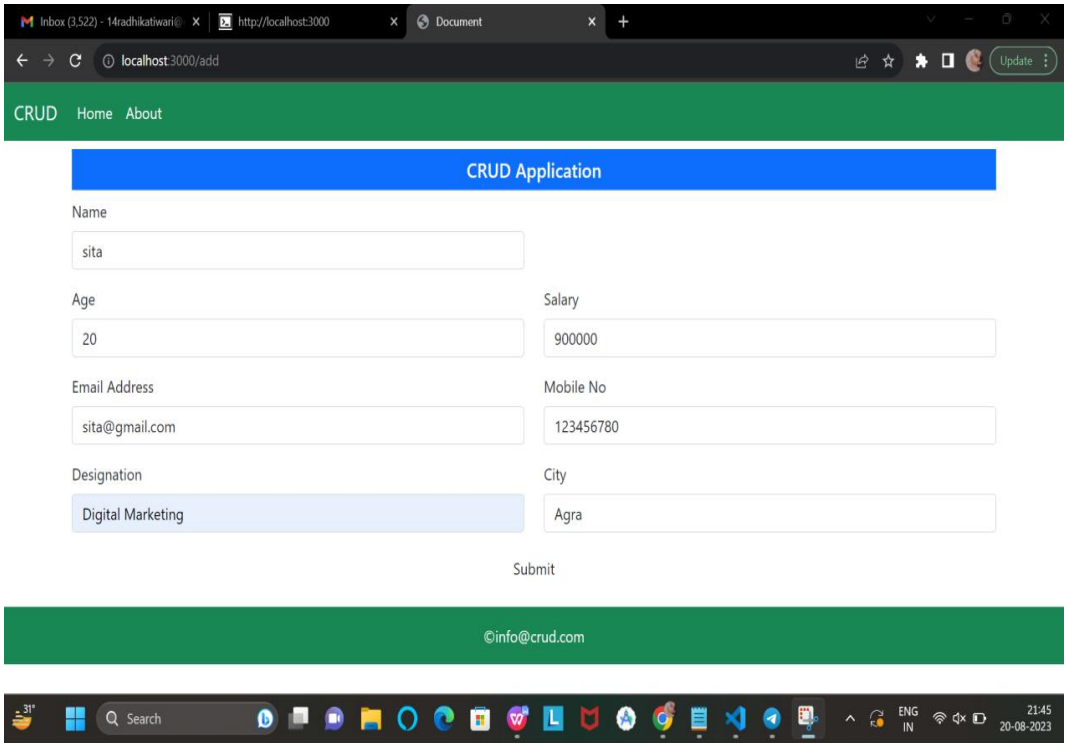


4. Testing

- **Invalid Data:**



- **Valid Data:**


Inbox (3,522) - 14radhikatiwari@
http://localhost:3000
Document
localhost:3000/add
Update
CRUD
Home
About
CRUD Application
Name
sita
Age
20
Salary
900000
Email Address
sita@gmail.com
Mobile No
123456780
Designation
Digital Marketing
City
Agra
Submit
©info@crud.com
Search
ENG
IN
21:45
20-08-2023

5. Implementation

5.1 HTML Structure:

The HTML file provides the structure for the application interface, including forms, buttons, and data display areas.

5.2 CSS Styling:

The CSS file is used to style the application, making it visually appealing and user-friendly.

5.3 JavaScript Functionality:

Create: Users can input data in a form and click a "Create" button to add an item to the list. JavaScript captures the input data, updates the list, and refreshes the display.

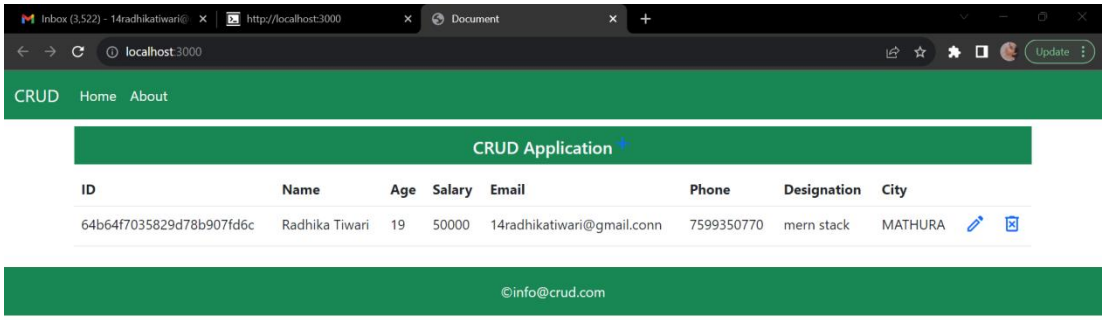
Read: The list of items is displayed on the page, allowing users to view the existing data.

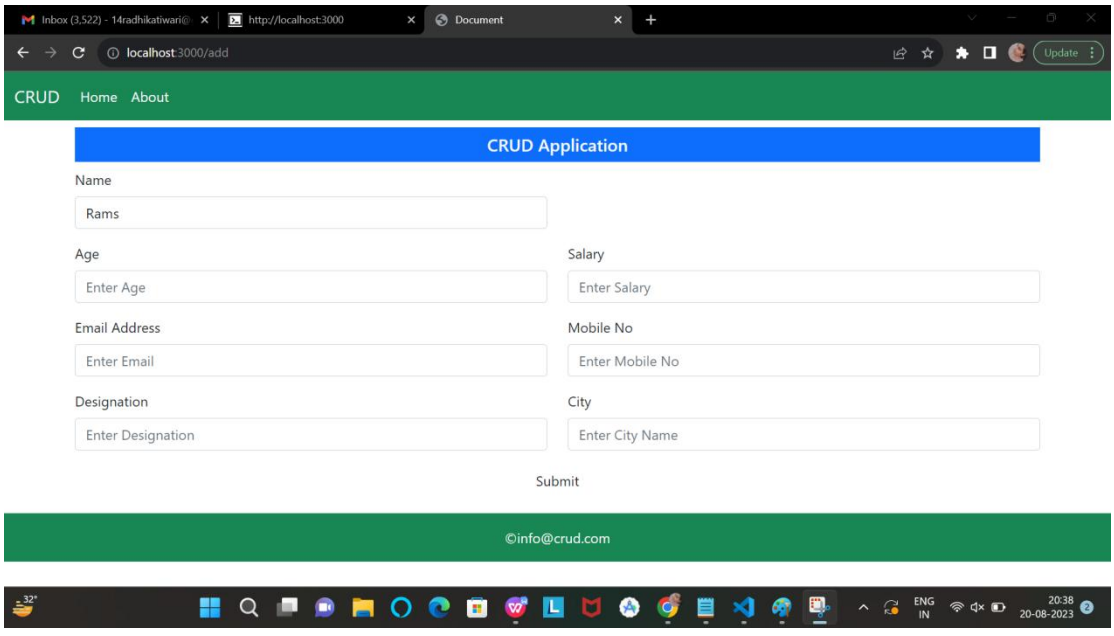
Update: Users can select an item, edit its details in a form, and click an "Update" button to save changes. JavaScript updates the data and refreshes the display.

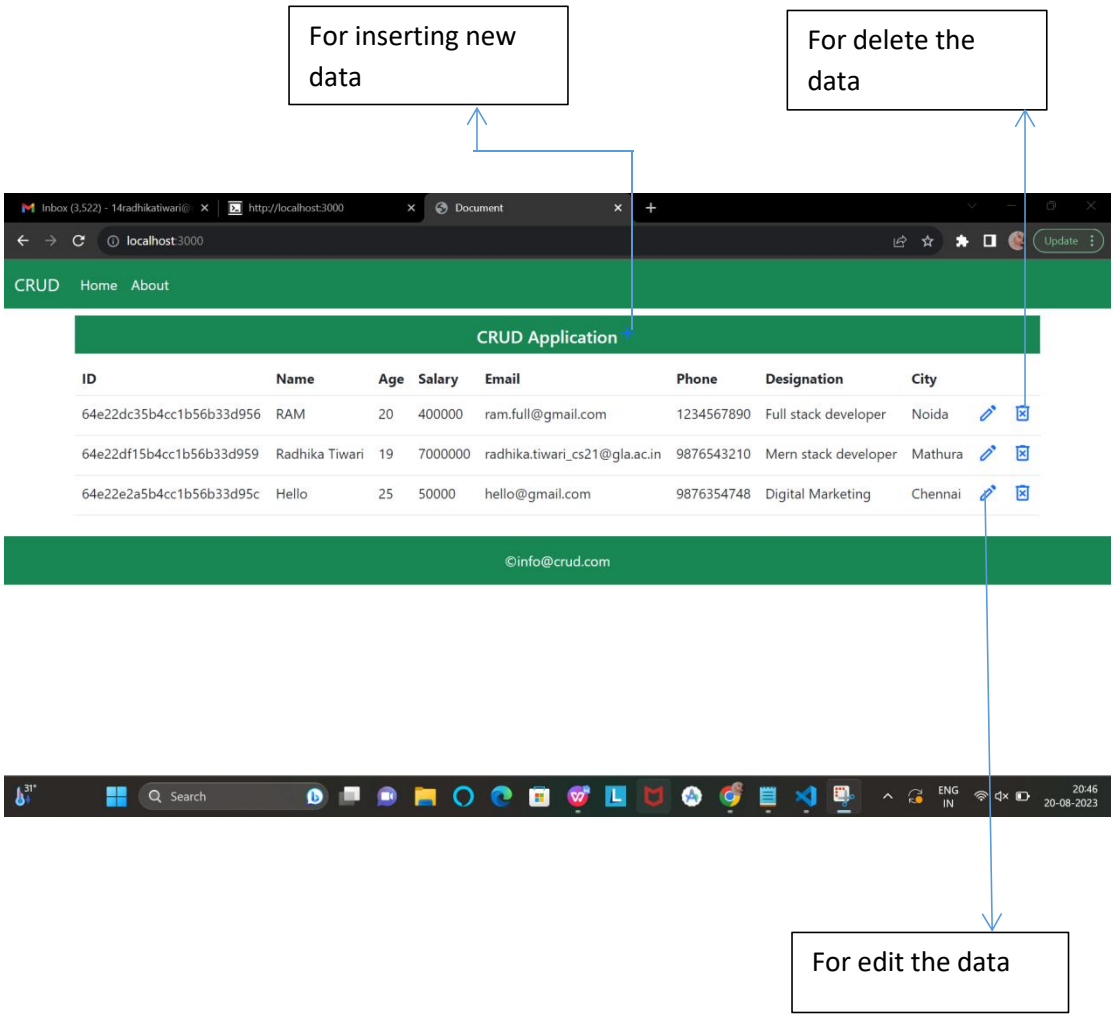
Delete: Each item in the list has a "Delete" button. When clicked, JavaScript removes the item from the list and updates the display.

5.4 Local Storage:

Data is stored locally using the browser's local Storage API to ensure that the data remains even after the page is closed.Data is loaded from local storage on page load and saved to local storage after each CRUD operation.


Inbox (3,522) - 14radhikatiwari@
http://localhost:3000
Document
localhost:3000
Update
CRUD
Home
About
CRUD Application
ID
Name
Age
Salary
Email
Phone
Designation
City
64b64f7035829d78b907fd6c
Radhika Tiwari
19
50000
14radhikatiwari@gmail.conn
7599350770
mern stack
MATHURA
©info@crud.com
ENG
IN
20:37
20-08-2023


Inbox (3,522) - 14radhikatiwari@
http://localhost:3000
Document
localhost:3000/add
Update
CRUD
Home
About
CRUD Application
Name
Rams
Age
Enter Age
Salary
Enter Salary
Email Address
Enter Email
Mobile No
Enter Mobile No
Designation
Enter Designation
City
Enter City Name
Submit
©info@crud.com
ENG
IN
20:38
20-08-2023


For inserting new data
For delete the data
localhost:3000
Update
CRUD
Home
About
CRUD Application
ID
Name
Age
Salary
Email
Phone
Designation
City
64e22dc35b4cc1b56b33d956
RAM
20
400000
ram.full@gmail.com
1234567890
Full stack developer
Noida
64e22df15b4cc1b56b33d959
Radhika Tiwari
19
7000000
radhika.tiwari_cs21@gla.ac.in
9876543210
Mern stack developer
Mathura
64e22e2a5b4cc1b56b33d95c
Hello
25
50000
hello@gmail.com
9876354748
Digital Marketing
Chennai
©info@crud.com
Search
ENG
IN
20:46
20-08-2023
For edit the data

References

- https://en.wikipedia.org/wiki/Servlet
- https://www.tutorialpoint.com/mysql/index.htm
- https://www.w3schools.com/java/java_intro.asp

6.Appendices

Employee.js

```
const mongoose=require('mongoose')
const EmployeesSchema=new mongoose.Schema({
   name:{
      type:String
   },
   age:{
     type:Number,
   },
   salary:{
     type:Number,
     default:30000,
  },
  email:{
   type:String
  },
  phone:{
   type:String
  },
  desig:{
    type:String
  },
  city:{
    type:String,
    default:"noida",
  },
```

```
})
const Employee= mongoose.model("Employee",EmployeesSchema)
module.exports=Employee
```

footer.hbs

```
<footer class="bg-success text-light text-center mt-2 p-3">
    &copy;info@crud.com
</footer>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGdlKrxdiJJigb/j/68SIy3Te4Bk
z" crossorigin="anonymous"></script>
</head>
<body></body>
```

header.hbs

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">

<link href="/css/style.css" rel="stylesheet">


<link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@20..48,100..700,0..1,-50..200" />
```

-

navbar.hbs

```
<nav class="navbar navbar-expand-lg bg-success">
  <div class="container-fluid">
    <a class="navbar-brand text-light" href="/">CRUD</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link text-light active" aria-current="page" href="/">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link text-light" href="#">About</a>
        </li>

      </ul>

    </div>
  </div>
</nav>
</head>
<body>
```

#style.css

```
.mytable{
   height: 390px;
   overflow-y: auto;
   overflow-x: hidden;
}
.icone
{
   color: white;
}
```

add.hbs

```
{{>header}}
<title>Add Record CodeSquadz</title>
{{>navbar}}
<div class="container">
   <h5 class="bg-primary text-light text-center mt-2 p-2">CRUD Application</h5>
   <form class="mytable" action="/add" method="post">
      <div class="row">
      <div class="mb-3 col-md-6 col-12">
         <label class="form-label">Name</label>
         <input type="text" class="form-control" name="name" placeholder="Enter
Name Here" />
      </div>
```

```
        </div>
        <div class="row">
        <div class="mb-3 col-md-6 col-12">
            <label class="form-label">Age</label>
            <input type="number" class="form-control" name="age" placeholder="Enter
Age" />
        </div>
        <div class="mb-3 col-md-6 col-12">
            <label class="form-label">Salary</label>
            <input type="number" class="form-control" name="salary"
placeholder="Enter Salary" />
        </div>
        </div>
        <div class="row">
        <div class="mb-3  col-md-6 col-12">
            <label class="form-label">Email Address</label>
            <input type="email" class="form-control" name="email" placeholder="Enter
Email"/>

        </div>

        <div class="mb-3 col-md-6 col-12">
            <label class="form-label">Mobile No</label>
            <input type="text" class="form-control" name="phone" placeholder="Enter
Mobile No" />
        </div>
        </div>
        <div class="row">
```

```
        <div class="mb-3 col-md-6 col-12">
            <label class="form-label">Designation</label>
            <input type="text" class="form-control" name="desig" placeholder="Enter
Designation" />
        </div>
        <div class="mb-3 col-md-6 col-12">
            <label class="form-label">City</label>
            <input type="text" class="form-control" name="city" placeholder="Enter City
Name" />
        </div>
        </div>
        <button type="submit" class="btn btn-priamry w-100">Submit</button>
    </form>



</div>
{{>footer}}
```

index.hbs

```
{{>header}}
    <title>All Record</title>
{{>navbar}}
<div class="container">
    <h5 class="bg-success text-light text-center mt-2 p-2">CRUD Application<a
href="/add"><span class="material-symbols-outlined">add</span></a></h5>
```

```
<div class="table-responsive">
<table class="table">
    <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Age</th>
        <th>Salary</th>
        <th>Email</th>
        <th>Phone</th>
        <th>Designation</th>
        <th>City</th>
        <th></th>
        <th></th>
    </tr>
    {{#each data}}
    <tr>
        <td>{{_id}}</td>
        <td>{{name}}</td>
        <td>{{age}}</td>
        <td>{{salary}}</td>
        <td>{{email}}</td>
        <td>{{phone}}</td>
        <td>{{desig}}</td>
        <td>{{city}}</td>
        <td><a href='/update/{{_id}}'><span class="material-symbols-
outlined">edit</span></a></td>
        <td><a href='/delete/{{_id}}'><span class="material-symbols-
outlined">delete_forever</span></a></td>
```

```
        </tr>
        {{/each}}
    </table>
    </div>
</div>
{{>footer}}
```

update.hbs

```
{{>header}}
<title>Update Record CodeSquadz</title>
{{>navbar}}
<div class="container">
    <h5 class="bg-success text-light text-center mt-2 p-2">CRUD Application</h5>
    <form class="mytable" action="/update/{{data._id}}" method="post">
        <div class="row">
        <div class="mb-3 col-md-6 col-12">
            <label class="form-label">Name</label>
            <input type="text" class="form-control" name="name"
value="{{data.name}}" />
        </div>
        </div>
        <div class="row">
        <div class="mb-3 col-md-6 col-12">
            <label class="form-label">Age</label>
```

```
          <input type="number" class="form-control" name="age"
value="{{data.age}}" />
      </div>
      <div class="mb-3 col-md-6 col-12">
          <label class="form-label">Salary</label>
          <input type="number" class="form-control" name="salary"
value="{{data.salary}}" />
      </div>
      </div>
      <div class="row">
      <div class="mb-3  col-md-6 col-12">
          <label class="form-label">Email Address</label>
          <input type="email" class="form-control" name="email"
value="{{data.email}}"/>

      </div>

      <div class="mb-3 col-md-6 col-12">
          <label class="form-label">Mobile No</label>
          <input type="text" class="form-control" name="phone"
value="{{data.phone}}" />
      </div>
      </div>
      <div class="row">
      <div class="mb-3 col-md-6 col-12">
          <label class="form-label">Designation</label>
          <input type="text" class="form-control" name="desig"
value="{{data.desig}}" />
      </div>
```

```
        <div class="mb-3 col-md-6 col-12">
            <label class="form-label">City</label>
            <input type="text" class="form-control" name="city" value="{{data.city}}"
/>
        </div>
        </div>
        <button type="submit" class="btn btn-success w-100">Submit</button>
    </form>



</div>
{{>footer}}
```

index.js

```
const express = require("express");
const hbs=require('hbs')

const Employee=require("./models/Employee")
const path=require('path')
const bodyParser=require('body-parser')

require("./dbConnect")
const app = express();
const encoder=bodyParser.urlencoded()
app.set("views","./views")
```

```
app.set("view engine","hbs")
const partialPath=path.join(__dirname,"./views/partials")
hbs.registerPartials(partialPath)
const staticPath=path.join(__dirname,"./views/public")
app.use(express.static(staticPath))

app.get("/",async(req,res)=>{
   var data=await Employee.find()
   res.render("index",{"data":data})
})
app.get("/add",(req,res)=>{
   res.render("add")
})

app.post("/add",encoder,async(req,res)=>
{
   var data=new Employee(req.body)
   await data.save();
   res.redirect("/")
})

app.get("/delete/:_id",async(req,res)=>
{
   await Employee.deleteOne({_id:req.params._id})
   res.redirect("/")
})
```

```
app.get("/update/:_id",async(req,res)=>{
   const data=await Employee.findOne({_id:req.params._id})
   res.render("update",{data:data})
})

app.post("/update/:_id",encoder,async(req,res)=>
{
   var data=await Employee.findOne({_id:req.params._id})
   data.name=req.body.name
   data.age=req.body.age
   data.salary=req.body.salary
   data.email=req.body.email
   data.phone=req.body.phone
   data.desig=req.body.desig
   data.city=req.body.city

   await data.save();
   res.redirect("/")
})

// Server listen

app.listen(3000, () => console.log("Server listening to port 3000"));
```