1. INTRODUCTION

1.1 Project Introduction

The “***BLOGHAVEN***” is developed to overcome the problems prevailing in the practicing manual system. This application is supported to eliminate and, in some case, reduce the hardships faced by this existing system. Moreover, this system is designed for the particular need of a single user to carry out operations in a smooth and effective manner.

The main challenges to overcome were managing the updated information of Blogs and Images. Every person likes to have different category of blogs which he/she would like to write on.

The application is reduced as much as possible to avoid errors while entering the data. No formal knowledge is needed for the user to use this system. Thus, by this all proves it is user friendly. It assists the user to concentrate on their productivity.

1.2 Purpose of the project

The purpose of this project is to create a blog application by the help of computerized equipment and full-fledged computer software, fulfilling their requirements so that their valuable data/information can be stored for a longer period with the easy accessing and manipulation of the same. The required software and hardware are easily available and easy to work with.

The project, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. That means that the user need not be distracted by the information that is not relevant, while being able to reach the information.

2. LITERATURE SURVEY

2.1 EXISTING SYSTEM

The existing system for online blogs includes traditional methods like just writing texts without images or comments, which are too slow and unpleasant to watch. With the advancement of internet, writing blogs online is very interesting and fun thing to do., which makes the blog writing a common thing now a days. Again, most of these are limited to the have technical skills, the writers are required to have the knowledge of the technologies for putting the logs online in the internet.

2.1.1 PROPOSED SYSTEM

BLOGHAVEN is developed to provide an effective means for the writers to post blogs to their page without any hassle. In addition, Writers can view and update their blogs, title and image at any time. This makes the application user-friendly. BLOGHAVEN is web-based application providing flexibility for the users.

2.2 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in the feasibility analysis are:

2.2.1 Economic Feasibility

This study is carried out to check the economic impact will have on the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system was within the budget and this was achieved because most of the technologies used are freely available. Only the customized products have to be purchased.

2.2.2 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes for the implementing this system.

2.2.3 Operational Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

2.3 TOOLS AND TECHNOLOGIES

MONGODB: [MongoDB: Cross-platform Document-Oriented Database]

MongoDB is a NoSQL database where each record is a document comprising of key-value pairs that are similar to JSON (JavaScript Object Notation) objects. MongoDB is flexible and allows its users to create schema, databases, tables, etc. Documents that are identifiable by a primary key make up the basic unit of MongoDB. Once MongoDB is installed, users can make use of Mongo shell as well. Mongo shell provides a JavaScript interface through which the users can interact and carry out operations (e.g.: querying, updating records, deleting records).

Why use MongoDB?

- Fast – Being a document-oriented database, easy to index documents. Therefore, a faster response.
- Scalability – Large data can be handled by dividing it into several machines.
- Use of JavaScript – MongoDB uses JavaScript which is the biggest advantage.
- Schema Less – Any type of data in a separate document.

Data stored in the form of JSON –

- Objects, Object Members, Arrays, Values, and Strings
- JSON syntax is very easy to use.
- JSON has a wide range of browser compatibility.
- Sharing Data: Data of any size and type (video, audio) can be shared easily.

Simple Environment Setup – It’s really simple to set up MongoDB.

Flexible Document Model – MongoDB supports document-model (tables, schemas, columns & SQL) which is faster and easier.

Express: Back-End Framework:

Express is a Node.js framework. Rather than writing the code using Node.js and creating loads of Node modules, Express makes it simpler and easier to write the back-end code. Express helps in designing great web applications and APIs. Express supports many middlewares which makes the code shorter and easier to write. Why use Express?

- Asynchronous and Single-threaded.
- Efficient, fast & scalable
- Has the biggest community for Node.js
- Express promotes code reusability with its built-in router.
- Robust API
- Create a new folder to start your express project and type below command in the command prompt to initialize a package.json file. Accept the default settings and continue.

React: Front-End Library

React is a JavaScript library that is used for building user interfaces. React is used for the development of single-page applications and mobile applications because of its ability to handle rapidly changing data. React allows users to code in JavaScript and create UI components. Why use React?

- Virtual DOM – A virtual DOM object is a representation of a DOM object. Virtual DOM is actually a copy of the original DOM. Any modification in the web application causes the entire UI to re-render the virtual DOM. Then the difference between the original DOM and this virtual DOM is compared and the changes are made accordingly to the original DOM.
- JSX – Stands for JavaScript XML. It is an HTML/XML JavaScript Extension which is used in React. Makes it easier and simpler to write React components.
- Components – ReactJS supports Components. Components are the building blocks of UI wherein each component has a logic and contributes to the overall UI. These components also promote code reusability and make the overall web application easier to understand.
- High Performance – Features like Virtual DOM, JSX and Components makes it much faster than the rest of the frameworks out there.
- Developing Android/iOS Apps – With React Native you can easily code Android-based or IOS-Based apps with just the knowledge of JavaScript and ReactJS.

Node.js: JS Runtime Environment

Node.js provides a JavaScript Environment which allows the user to run their code on the server (outside the browser). Node pack manager i.e., npm allows the user to choose from thousands of free packages (node modules) to download. Why use Node.JS?

- Open-source JavaScript Runtime Environment
- Single threading – Follows a single-threaded model.
- Data Streaming
- Fast – Built on Google Chrome’s JavaScript Engine, Node.js has a fast code execution.
- Highly Scalable
- Initialize a Node.js application by typing running the below command in the command window. Accept the standard settings.

Advantages of MERN Stack

There are a lot of advantages of MERN Stack, some of them are mentioned below -

- For a smooth development of any web application or mobile app, it supports MVC (Model View Controller) architecture; the main purpose of this architecture is to separate the presentation details with the business logic.
- It covers all the web development stages starting from front-end development to backend development with JavaScript.
- It is an open-source framework mainly used to develop web-based or mobile applications and is supported by the community.

3. SOFTWARE AND HARDWARE REQUIREMENTS

3.1 INTRODUCTION

To be used efficiently, all computer software needs certain hardware components or the other software resources to be present on a computer. These pre-requisites are known as(computer) system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements.

3.2 SYSTEM REQUIREMENTS

3.2.1 HARDWARE REQUIREMENTS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatibility and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

HARDWARE REQUIREMENTS FOR PRESENT PROJECT:

PROCESSOR: Intel i3 or higher

RAM: 4 GB

HARD DISK: 80 GB

3.2.2 SOFTWARE REQUIREMENTS:

Software Requirements deal with defining software resource requirements and pre-requisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or pre-requisites are generally not included in the software installation package and need to be installed separately before the software is installed.

SOFTWARE REQUIREMENTS FOR PRESENT PROJECT:

OPERATING SYSTEM: Windows 8 or higher / Linux (Ubuntu 20.04 LTS)

FRONT-END: ReactJS, Material-UI (Framework)

SERVER-SIDE SCRIPT: NodeJS, ExpressJS (Framework)

DATABASE: mongo DB

3.3 MODULES

This project has the following modules: -

3.3.1 BLOG MANAGEMENT MODULE

- The user can create a blog with title, description and an image.
- The user can update the content of the blog along with the title and image.
- The user can delete the blog if it is of no longer use.

3.3.2 CATEGORY MANAGEMENT MODULE

- There are multiple categories from which the user can select the genre of his/her choice.
- The categories are displayed in a lexicographical order.
- The blogs can be filtered out according to the categories selected on the home page.

3.3.3 COMMENT MANAGEMENT MODULE

- This module handles the comments posted on the blog.
- The comments posted in the blog gets stored in the database and gets updated in real time on the browser.
- The comments can be deleted from the blog and it is reflected back to the database.

3.3.4 IMAGE MANAGEMENT MODULE

- This module manages the images uploaded to the website.
- The images when gets uploaded on the database, it gets converted in the forms of chunks to save the bandwidth and the space.
- The image after getting uploaded in the database, it again gets converted back to the image from the chunks and gets displayed back on the browser.

4 SYSTEM DESIGN

4.1 INTRODUCTION TO UML:

4.1.1 UML Design

The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the software system and its components. It is a graphical language, which provides a vocabulary and set of semantics and rules. The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that must be constructed. It is used to understand, design, configure, maintain, and control information about the systems.

The UML is a language for:

• Visualizing

• Specifying

• Constructing

• Documenting

Visualizing

Through UML we see or visualize an existing system and ultimately, we visualize how the system is going to be after implementation. Unless we think, we cannot implement. UML helps to visualize, how the components of the system communicate and interact with each other.

Specifying

Specifying means building, models that are precise, unambiguous and complete UML addresses the specification of all the important analysis design, implementation decisions that must be made in developing and deploying a software system.

Constructing

UML models can be directly connected to a variety of programming language through mapping a model from UML to a programming language like JAVA or C++ or VB. Forward Engineering and Reverse Engineering is possible through UML.

Documenting

The Deliverables of a project apart from coding are some Artifacts, which are critical in controlling, measuring and communicating about a system during its developing requirements, architecture, desire, source code, project plans, tests, prototypes releasers, etc.

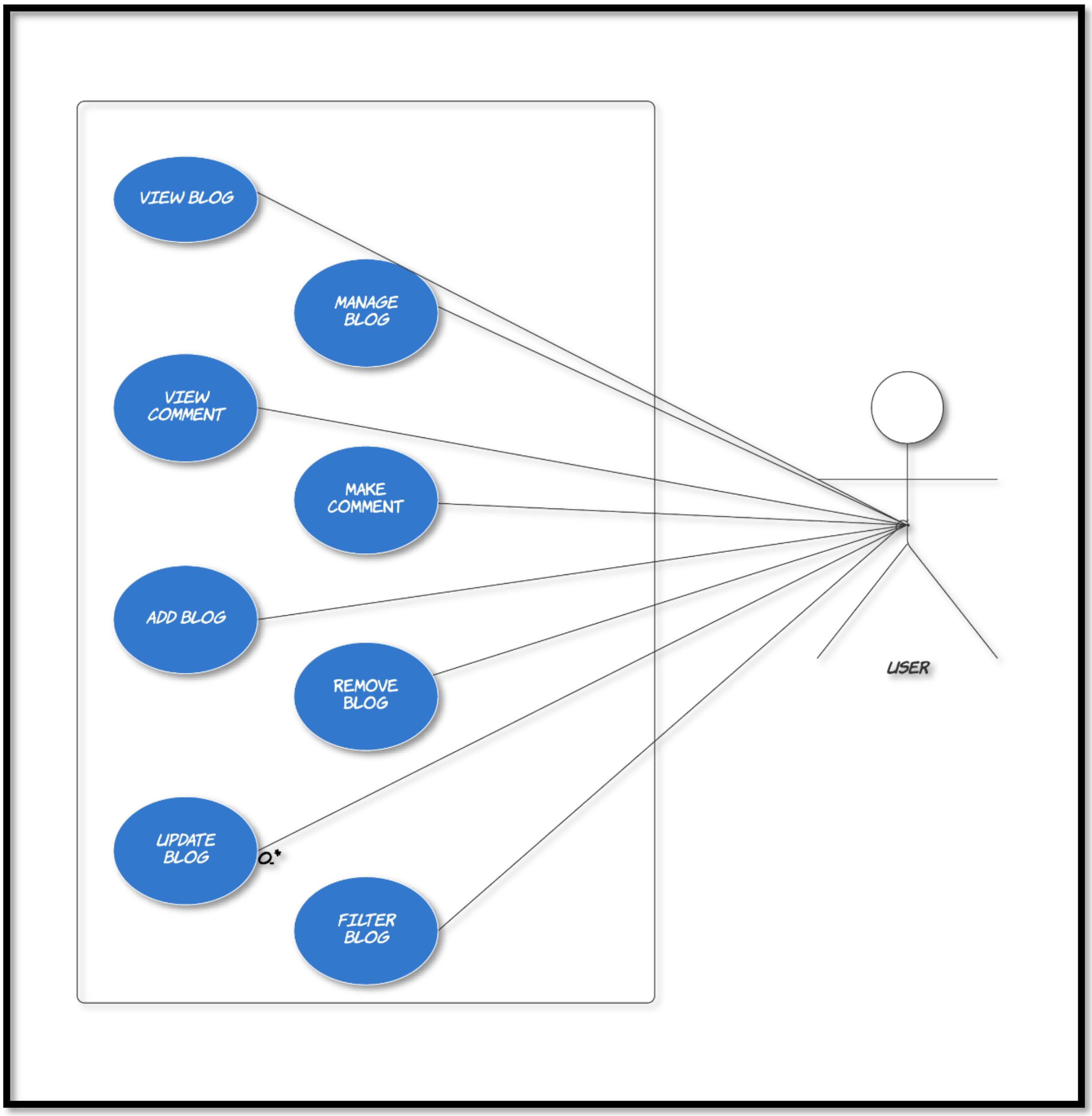
4.1 USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is type of behavioral diagram defined by and created from a use-case analysis. its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

Use case diagrams are formally included in two modeling languages defined by the OMG:

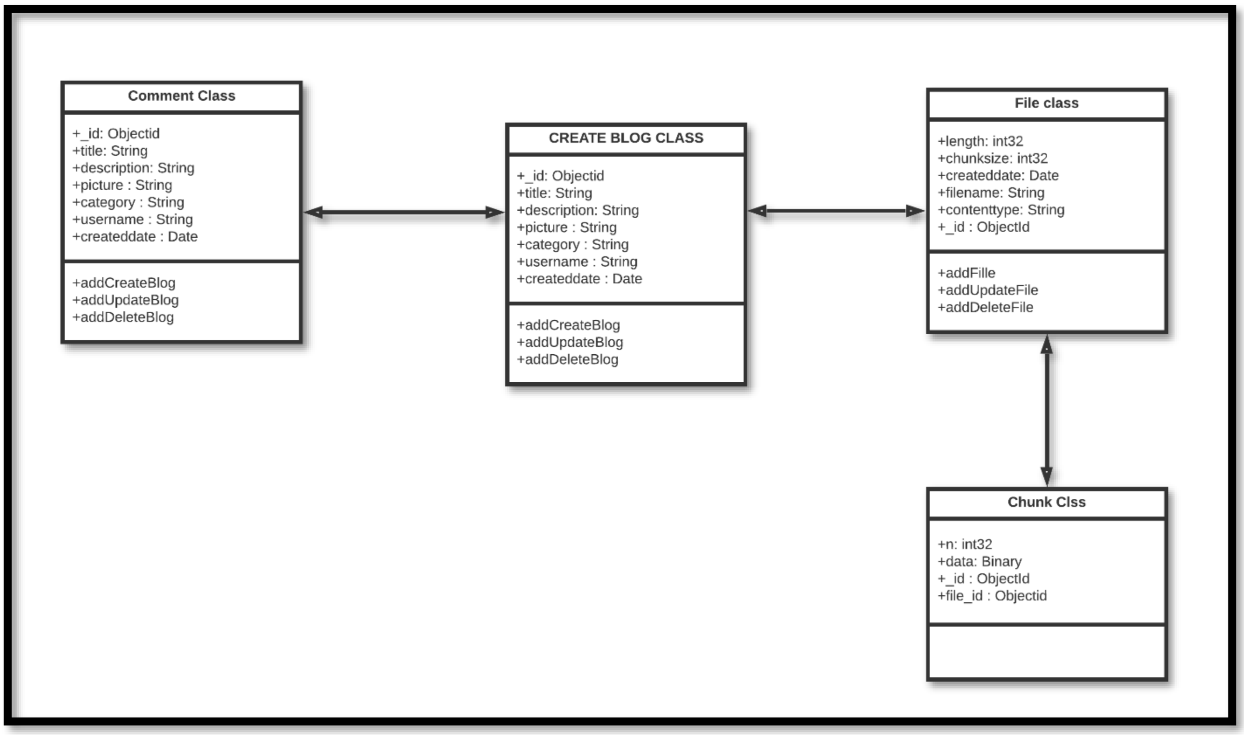
the unified modeling language (UML) and the systems modeling language (sys ML)

Use case diagram of our project: Use Case Diagram: User

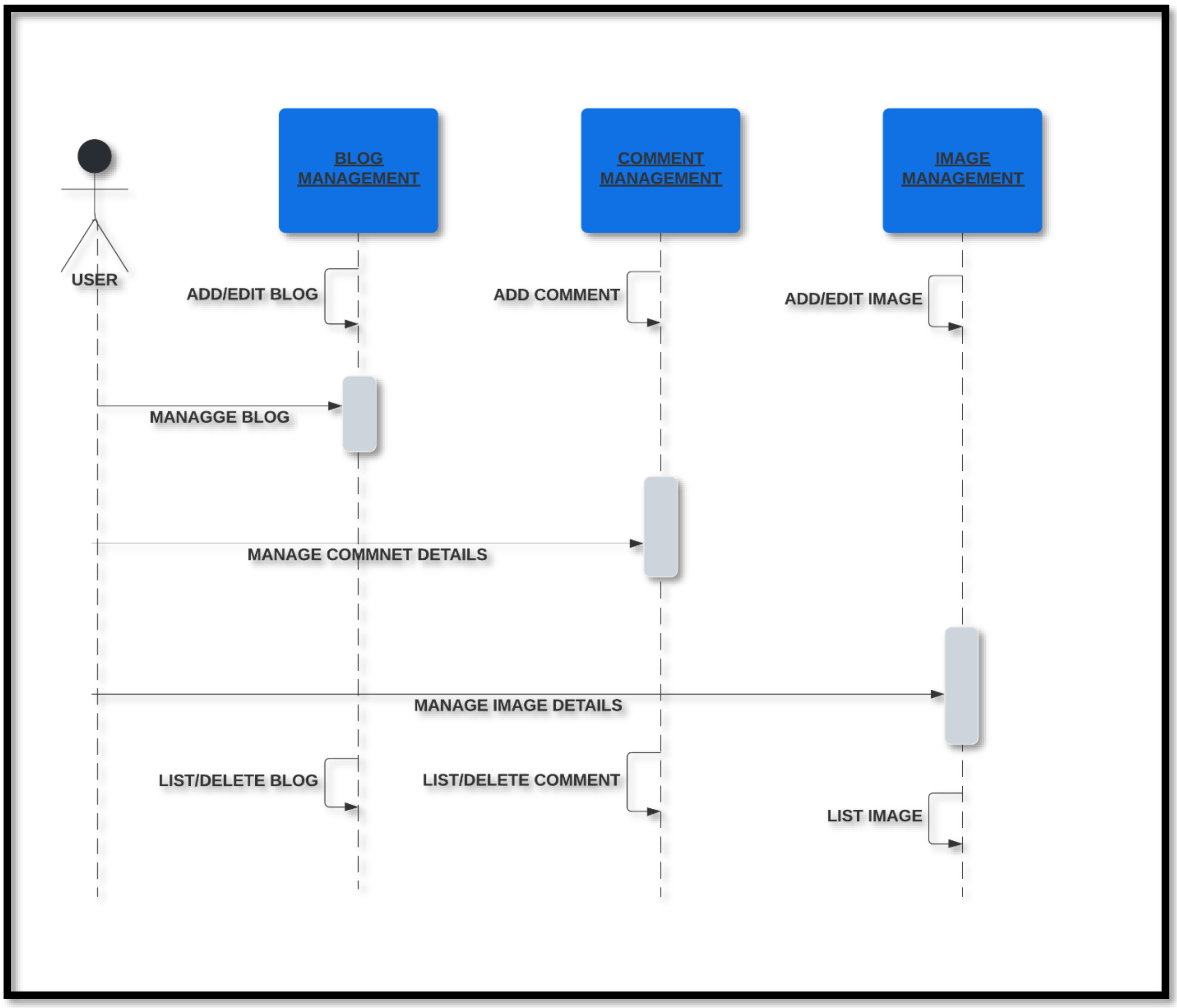


4.2 Class Diagram:

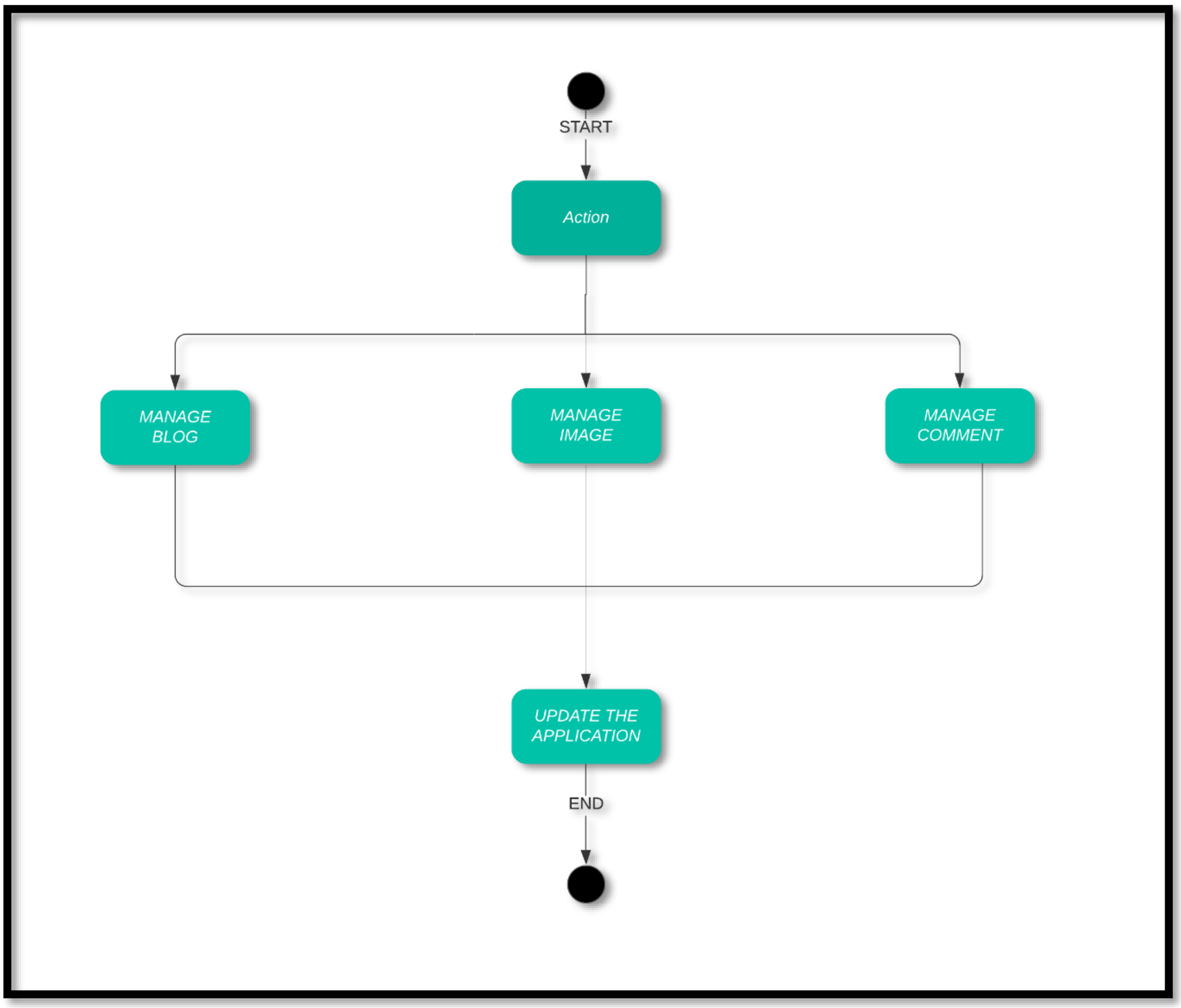
A Class is a category or group of things that has similar attributes and common behavior. A Rectangle is the icon that represents the class it is divided into three areas. The upper most area contains the name, the middle; area contains the attributes and the lowest areas show the operations. Class diagrams provides the representation that developers work from. Class diagrams help on the analysis side, too.



4.3 Sequence Diagram



4.4 Activity Diagram

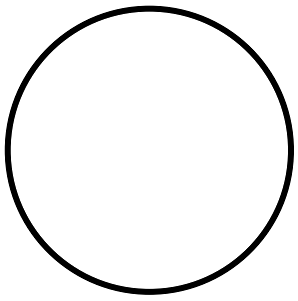


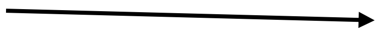
4.5 Data flow diagram

A DFD does not show a sequence of steps. A DFD only shows what the different process in a system is and what data flows between them.

The following are some DFD symbols used in the project.

External Entities


Process: A transaction of information that resides within the bounds of the system to be module


Dataflows

DATASTORE: A repository of data that is to be stored for use by one or more processes, may be as simple as buffer of queue or as a relational database.

RULES FOR DFD:

• Fix the scope of the system by means of context diagrams.

• Organize the DFD so that the main sequence of the actions reads left to right and top to bottom.

• Identify all inputs and outputs.

• Identify and label each process internal to the system with rounded circles.

• A process is required for all the data transformation and transfers. Therefore, never connect a data store to a data source or the destinations or another data store with just a data flow arrow.

• Do not indicate hardware and ignore control information.

• Make sure the names of the processes accurately convey everything the process is done.

• There must not be unnamed process.

• Indicate external sources and destinations of the data, with squares.

• Number each occurrence of repeated external entities.

• Identify all data flows for each process step, except simple Record retrievals.

• Label data flow on each arrow.

• Use details flow on each arrow.

• Use the details flow arrow to indicate data movements.

• There can't be unnamed data flow.

• A data flow can't connect two external entities.

LEVELS OF DFD:

The complexity of the business system means that it is a responsible to represent the operations of any system of single data flow diagram. At the top level, an Overview of the different systems in an organization is shown by the way of context analysis diagram. When exploded into DFD

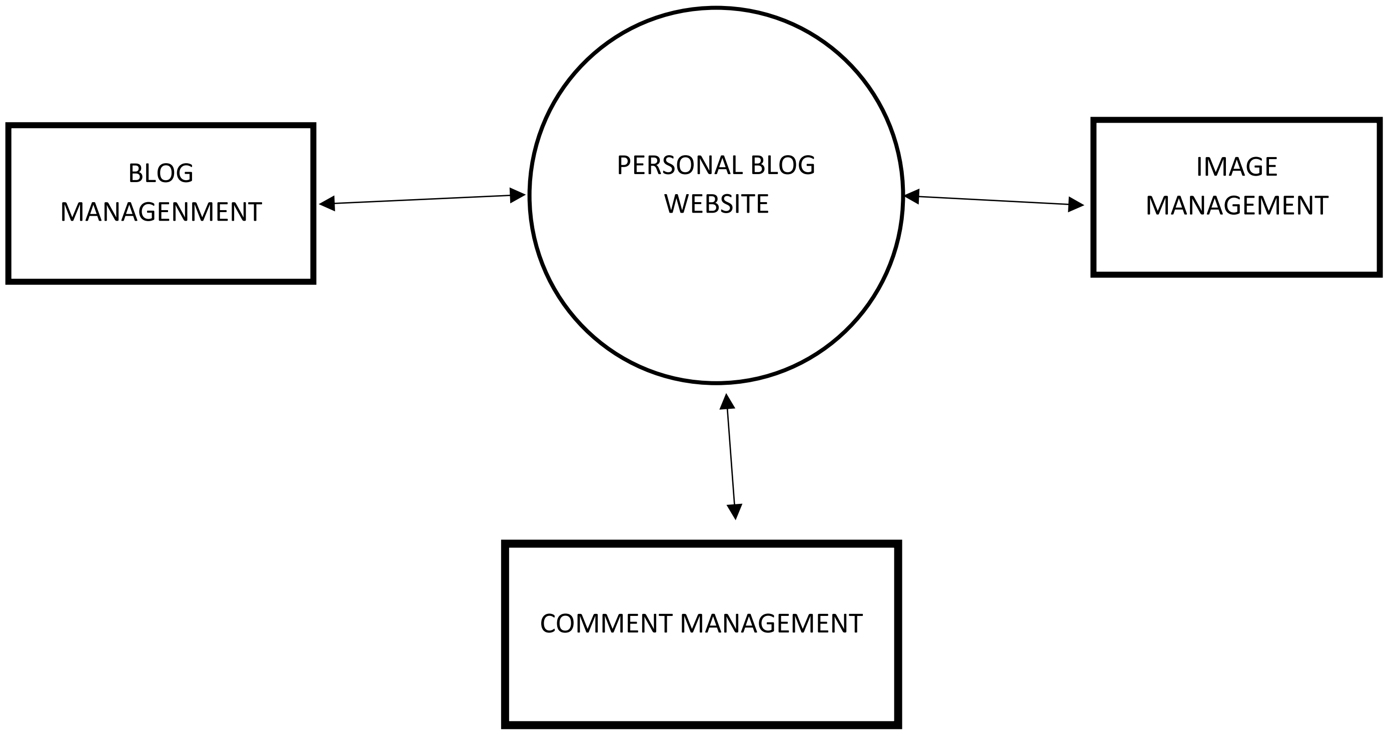
They are represented by:

- LEVEL-0: SYSTEM INPUT/OUTPUT
- LEVEL-1: SUBSYSTEM LEVEL DATAFLOW FUNCTIONAL
- LEVEL-2: FILE LEVEL DETAIL DATA FLOW.

The input and output data shown should be consistent from one level to the next.

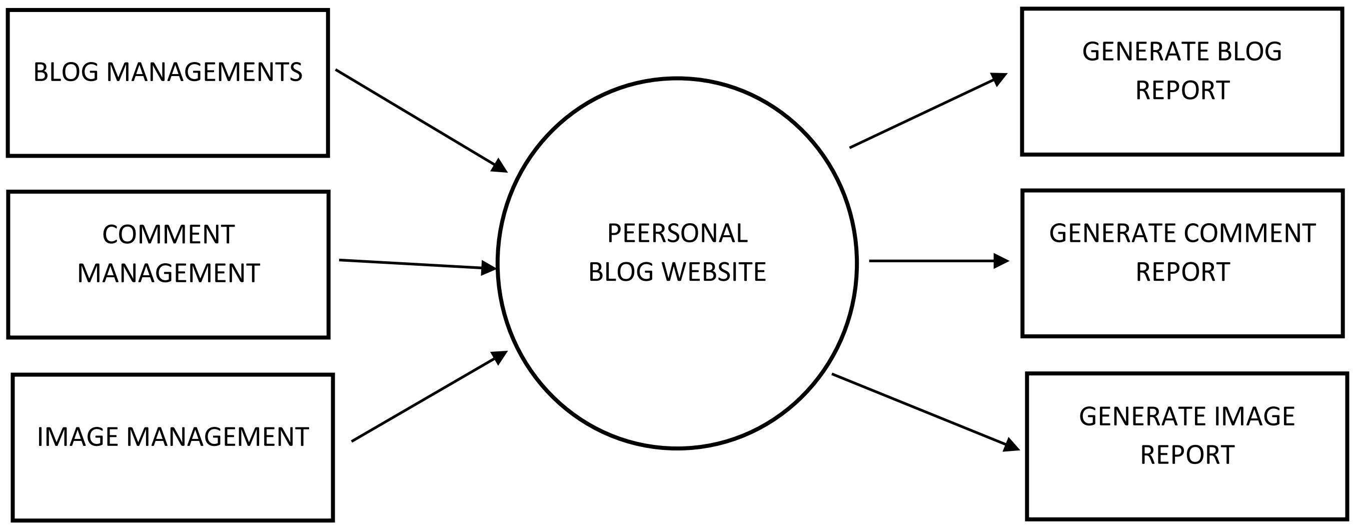
LEVEL-0: SYSTEM INPUT/OUTPUT LEVEL

A level-0 DFD describes the system-wide boundaries, dealing inputs to and outputs from the system and major processes. This diagram is similar to the combined user-level context diagram.



LEVEL-1: SUBSYSTEM LEVEL DATA FLOW

A level-1 DFD describes the next level of details within the system, detailing the data flows between subsystems, which makeup the whole.

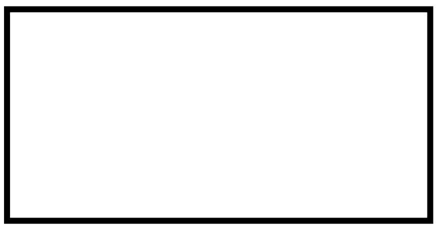
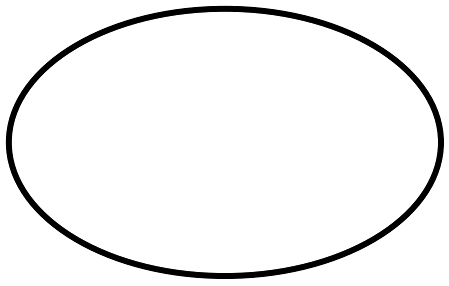
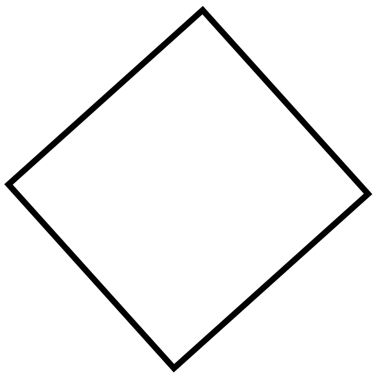
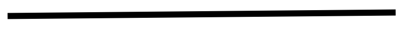


4.6 ER DIAGRAM

ENTITY-RELATIONSHIP Diagrams

E-R (Entity-Relationship) Diagram is used to represents the relationship between entities in the table.

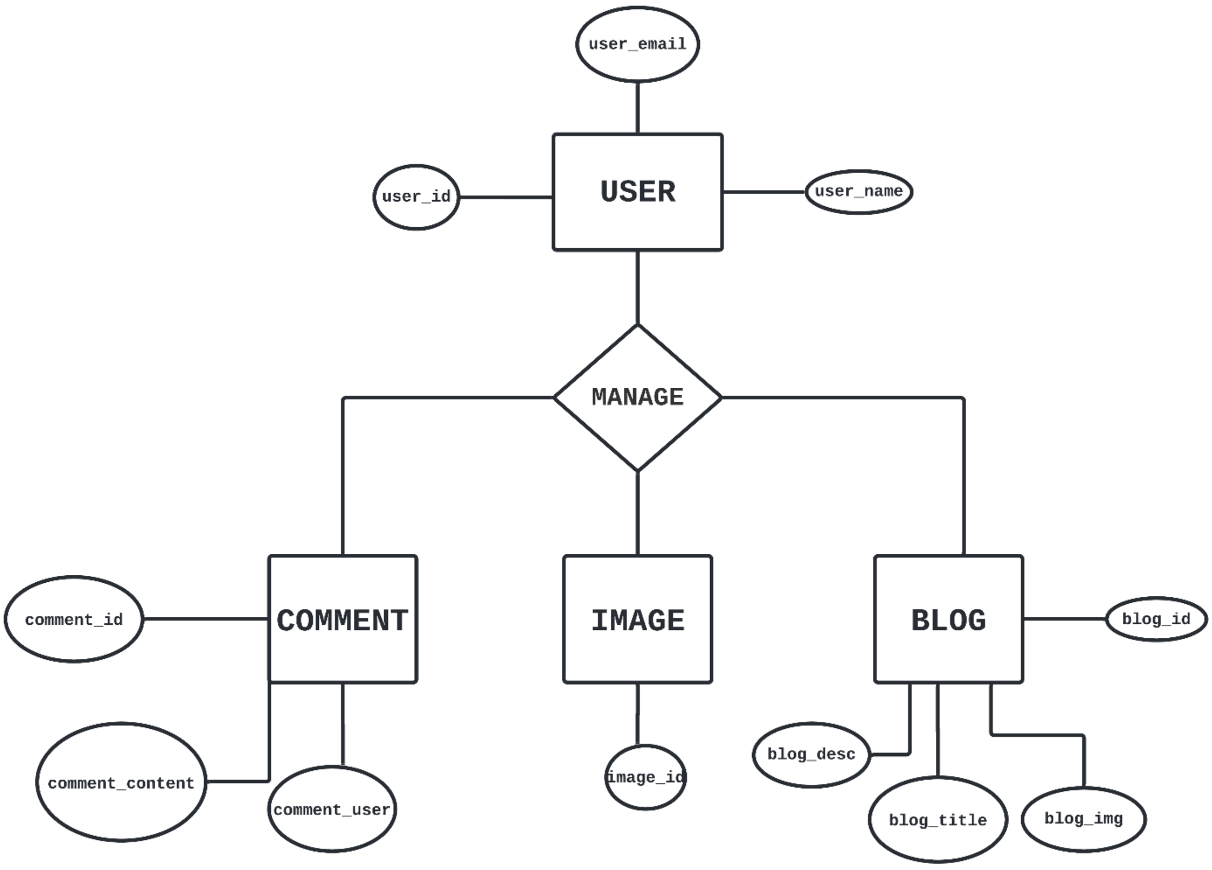
The symbols used in E-R diagrams are:

SYMBOLS	**PURPOSE**
	Represents Entity Set
	Represent attributes
	Represents Relationship Set
	Represents Line flow

Structured analysis is a set of tools and techniques that the analyst.

To develop a new kind of a system:

The traditional approach focuses on the cost benefit and feasibility analysis, Project management, and hardware and software selection a personal consideration.



5. IMPLEMENTATION

5.1 Introduction:

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus, it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

5.2 Codes

5.2.1 Index.html [client / public / index.html]

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <meta name="theme-color" content="#000000" />
  <meta name="description" content="Web site created using create-react-app" />
  <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
  <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link href=https://fonts.googleapis.com/css2?family=Dongle&family=Special+Elite&family=Ubuntu&
display= wap    rel="stylesheet">
  <title>Blog Website</title>
</head>
<body>
  <noscript>You need to enable JavaScript to run this app.</noscript>
  <div id="root"></div>
</body>
</html>
```

5.2.2 About.jsx [client/scr/component/about/about.jsx]

```
import { Box, makeStyles, Typography } from "@material-ui/core";


const useStyles = makeStyles({
  banner: {
    backgroundImage: `url(${"https://www.wallpapertip.com/wmimgs/23-236943_us-wallpaper-for-
website.jpg"})`,
    width: "100%",
    height: "50vh",
    backgroundPosition: "left 0px bottom 0px",
    backgroundSize: "cover",
  },
  wrapper: {
    padding: 20,
    "& > *": {
      marginTop: 50,
    },
  },
  text: {
    color: "#878787",
  },
});


const About = () => {
  const classes = useStyles();
  return (
    <Box>
      <Box className={classes.banner}></Box>
      <Box className={classes.wrapper}>
        <Typography variant="h3">Rihan</Typography>
        <Typography variant="h5" className={classes.text}>
          I am a student of Acharya Institute of Graduate Studies,
          Bangalore, India.
          <br />
          <br />
          “BLOGHAVEN” is developed to overcome the problems prevailing
          in the practicing manual system. This application is supported to
          eliminate and, in some cases, reduce the hardships faced by this
          existing system. Moreover, this system is designed for the particular
          need of a single user to carry out operations in a smooth and
          effective manner. <br />
          <br />
          The main challenges to overcome were managing the updated information
          of Blogs and Images. Every person likes to have different category of
          blogs which he/she would like to write on.
          <br />
          <br />
          The application is reduced as much as possible to avoid errors while
          entering the data. No formal knowledge is needed for the user to use
          this system. Thus, by this all proves it is user friendly. It assists
          the user to concentrate on their productivity.
        </Typography>
      </Box>
    </Box>
  );
};

export default About;
```

5.2.3 Comment.jsx [client/scr/component/commnets/comment.jsx]

```
import { Typography, Box, makeStyles } from "@material-ui/core";
import { Delete } from '@material-ui/icons';
import { deleteComment } from '../../services/api';
const useStyles = makeStyles({
    component: {
        marginTop: 30,
        background: '#F5F5F5',
        padding: 10
    },
    container: {
        display: 'flex',
        marginBottom: 5
    },
    name: {
        fontWeight: 600,
        fontSize: 18,
        marginRight: 20
    },
    date: {
        fontSize: 14,
        color: '#878787'
    },
    delete: {
        marginLeft: 'auto',
        cursor:'pointer'
    }
})
const Comment = ({ comment, setToggle }) => {
    const classes = useStyles();
    const removeComment = async () => {
       await deleteComment(comment._id) ;
       setToggle(prev => !prev);
    }
    return (
        <Box className={classes.component}>
            <Box className={classes.container}>
                <Typography className={classes.name}>{comment.name}</Typography>
                <Typography className={classes.date}>{new
Date(comment.date).toDateString()}</Typography>
                <Delete className={classes.delete} onClick={() => removeComment()} />
            </Box>
            <Typography>{comment.comments}</Typography>
        </Box>
    )
```

```
}
export default Comment;
```

5.2.4 Comments.jsx [client/scr/component/commnets/comments.jsx]

```
import { useState, useEffect } from 'react';
import { Box, TextareaAutosize, Button, makeStyles } from '@material-ui/core';
import { newComment, getComments } from '../../services/api';
//components
import Comment from './comment';
const useStyles = makeStyles({
    container: {
        marginTop: 60,
        display: 'flex',
        '& > *': {
            padding: '10px '
        }
    },
    image: {
        width: 50,
        height: 50,
        borderRadius: '50%'
    },
    textarea: {
        height: 100,
        width: '100%',
        margin: '0 20px'
    },
    button: {
        height: 40
    }
})
const initialValue = {
    name: '',
    postId: '',
    date: new Date(),
    comments: ''
}
const Comments = ({ post }) => {
    const classes = useStyles();
    const url = 'https://static.thenounproject.com/png/12017-200.png'
    const [comment, setComment] = useState(initialValue);
    const [comments, setComments] = useState([]);
    const [data, setData] = useState();
    const [toggle, setToggle] = useState(false);
    useEffect(() => {
        const getData = async () => {
            const response = await getComments(post._id);
            setComments(response);
        }
        getData();
    }, [toggle, post]);
    const handleChange = (e) => {
        setComment({
            ...comment,
            name: post.username,
            postId: post._id,
            comments: e.target.value
        });
        setData(e.target.value);
    }
    const addComment = async() => {
        await newComment(comment);
        setData('')
        setToggle(prev => !prev);
    }
    console.log(post);
    return (
        <Box>
```

```
            <Box className={classes.container}>
                <img src={url} className={classes.image} alt="dp" />
                <TextareaAutosize
                    rowsMin={5}
                    className={classes.textarea}
                    placeholder="What's on your mind?"
                    onChange={(e) => handleChange(e)}
                    value={data}
                />
                <Button
                    variant="contained"
                    color="primary"
                    size="medium"
                    className={classes.button}
                    onClick={(e) => addComment(e)}
                >Post</Button>
            </Box>
            <Box>
                {
                    comments && comments.map(comment => (
                        <Comment comment={comment} setToggle={setToggle} />
                    ))
                }
            </Box>
        </Box>
    )
}
export default Comments;
```

5.2.5 Contact.jsx [client/scr/component/contact/contact.jsx]

```
import { Box, makeStyles, Typography, Link } from "@material-ui/core";
import { GitHub, Instagram, Email } from "@material-ui/icons";


const useStyles = makeStyles({
  banner: {
    backgroundImage: `url(${"http://mrtaba.ir/image/bg2.jpg"})`,
    width: "100%",
    height: "50vh",
    backgroundPosition: "left 0px top -100px",
    backgroundSize: "cover",
  },
  wrapper: {
    padding: 20,
    "& > *": {
      marginTop: 50,
    },
  },
  text: {
    color: "#878787",
  },
});

const Contact = () => {
  const classes = useStyles();
  return (
    <Box>
      <Box className={classes.banner}></Box>
      <Box className={classes.wrapper}>
        <Typography variant="h3">Getting in touch is easy!</Typography>
        <Typography variant="h5" className={classes.text}>
          Need something built or simply want to have chat? Reach out to me on
          <Box component="span" style={{ marginLeft: 5 }}>
            <Link
              href="https://www.instagram.com/raihan.exe_isnotworking"
              color="inherit"
              target="_blank"
            >
              <Instagram />
            </Link>
          </Box>
          or send me an Email
          <Link
            href="mailto:rihanhassan4@gmail.com"
            target="_blank"
            color="inherit"
          >
            <Email />
          </Link>
          . If you are interested, you can view my portfolio here
          <Box component="span" style={{ marginLeft: 5 }}>
            <Link
              href="https://github.com/Raihanhassan10"
              color="inherit"
              target="_blank"
            >
              <GitHub /> - Rihan
            </Link>
          </Box>
        </Typography>
      </Box>
    </Box>
  );
};

export default Contact;
```

5.2.6 Banner.jsx [client/scr/component/home/banner.jsx]

```
import { makeStyles, Box, Typography } from "@material-ui/core";
const Styles = makeStyles((theme) => ({
  image: {
    background: `url(/peakpx.jpg) #000`,
    width: "100%",
    height: "35vh",
    display: "flex",
    flexDirection: "column",
    alignItems: "center",
    justifyContent: "center",
    "& :first-child": {
      fontSize: 80,
      fontFamily: 'Special Elite,cursive',
      color: "black",
      [theme.breakpoints.down("sm")]: {
        // lineHeight: 1,
        fontSize: 40,
      },
    },
  },
}));
const Banner = () => {
  const classes = Styles();
  // const url = 'https://images.pexels.com/photos/768474/pexels-photo-768474.jpeg'
  return (
    <Box className={classes.image}>
      <Typography>BLOGHAVEN</Typography>
      {/* <Typography>| 5th Semester Project |</Typography> */}
    </Box>
  );
};
export default Banner;
```

5.2.7 Categories.jsx [client/scr/component/home/categories]

```
import {
  Button,
  makeStyles,
  Table,
  TableHead,
  TableCell,
  TableRow,
  TableBody,
} from "@material-ui/core";
import { categories } from "../../constants/data";
import { Link } from "react-router-dom";


const Styles = makeStyles({
  create: {
    margin: 15,
    background: "#b82e1f",
    color: "#ffffff",
    width: "80%",
  },
  table: {
    border: "1px solid rgba(224,224,224,1)",
  },
  link: {
    textDecoration: "none",
    color: "inherit",
  },
});
const Categories = () => {
  const classes = Styles();
  return (
    <>
      <Link to="/create" className={classes.link}>
        <Button variant="contained" className={classes.create}>
          Create Blog
        </Button>
      </Link>
      <Table className={classes.table}>
        <TableHead>
          <TableRow>
            <TableCell>
              <Link to={"/"} className={classes.link}>
                All Categories
              </Link>
            </TableCell>
          </TableRow>
        </TableHead>
        <TableBody>
          {categories.map((category) => (
            <TableRow>
              <TableCell>
                <Link to={`/?category=${category}`} className={classes.link}>
                  {category}
                </Link>
              </TableCell>
            </TableRow>
          ))}
        </TableBody>
      </Table>
    </>
  );
};
export default Categories;
```

5.2.8 Home.jsx [client/scr/component/home/home]

```
// Components import
import { Grid } from "@material-ui/core";
import Banner from "./Banner";
import Categories from "./Categories";
import Posts from "./Posts";
const Home = () => {
  return (
    <>
      <Banner />
      <Grid container>
        <Grid item lg={2} sm={2} xs={12}>
          <Categories />
        </Grid>
        <Grid container item lg={10} sm={10} xs={12}>
          <Posts />
        </Grid>
      </Grid>
    </>
  );
};
export default Home;
```

6.2.9 Post.jsx [client/scr/component/home/post]

```
import { Box, Typography, makeStyles } from "@material-ui/core"
const Styles = makeStyles({
  container:{
    height:400,
    margin: 10,
    border:'1px solid #d3cede',
    borderRadius:10,
    display:'flex',
    flexDirection:'column',
    alignItems:'center',
    '& > *':{
      padding: '0px 5px 5px 5px'
    }
  },
  image:{
    borderRadius:'10px 10px 0px 0px',
    height:150,
    width:'100%',
    objectFit:'cover'
  },
  text :{
    fontSize: 14,
    fontWeight:550,
    color: 'grey'
  },
  heading:{
    fontSize:19,
    fontWeight:600,
    color:'black',
    textAlign:'center'
  },
  details:{
    fontSize:14,
    fontWeight:550,
    color:'#2A272E',
    wordBreak: 'break-word'
  }
})
const Post = ({ post }) => {
  const classes = Styles();
  const url = post.picture || 'https://images.pexels.com/photos/5077047/pexels-photo-5077047.jpeg';
  const addElipsis = (str,limit) => {
    return str.length > limit ? str.substring(0, limit) + ' ...' : str;
  }
  return (
    <Box className= {classes.container}>
      <img src={url} alt='Bannerimage' className= {classes.image}/>
      <Typography className= {classes.text}>{post.categories}</Typography>
      <Typography className= {classes.heading}>{addElipsis(post.title,25)}</Typography>
      <Typography className= {classes.text}>Author: {post.username}</Typography>
      <Typography className= {classes.details}>{addElipsis(post.description,200)}</Typography>
    </Box>
  );
};
export default Post;
```

5.2.10 Posts.jsx [client/scr/component/home/posts]

```
import { Grid, Box } from "@material-ui/core";
import { useEffect, useState } from "react";
import { Link, useLocation } from "react-router-dom";
import { getAllPosts } from "../../services/api";
import Post from "./Post";
const Posts = () => {
  const [posts, setPosts] = useState([]);
  const { search } = useLocation();
  useEffect(() => {
    const fetchData = async () => {
      let data = await getAllPosts(search);
      console.log(data);
      setPosts(data);
    };
    fetchData();
  }, [search]);
  return (
    <>
      {posts.length ? (
        posts.map((post) => (
          <Grid item lg={3} sm={4} xs={12}>
            <Link
              style={{ textDecoration: "none", color: "inherit" }}
              to={`/details/${post._id}`}
            >
              <Post post={post} />
            </Link>
          </Grid>
        ))
      ) : (
        <Box style={{ color: "878787", margin: "30px 80px", fontSize: 28 }}>
          No data is available for this category.
        </Box>
      )}
    </>
  );
};
export default Posts;
```

5.2.11 DetailedView.jsx [client/scr/component/Post/DetailedView.jsx]

```
import { Box, makeStyles, Typography } from "@material-ui/core";
import { Edit, Delete}from '@material-ui/icons';
import { useEffect , useState } from "react";
import { Link, useHistory} from "react-router-dom";
import { getPost, deletePost } from "../../services/api";
import Comments from '../Comments/comments';

const Styles=makeStyles((theme) =>({
  container:{
    padding: '0px 100px',
    [theme.breakpoints.down('md')]:{
      padding:0
    }
  },
  image:{
    width: '100%',
    height:'35vh',
    objectFit: 'cover'
  },
  icons:{
    float:'right'
  },
  icon:{
    margin: 5,
    border: '1px solid #878787',
    padding:  5,
    borderRadius:10
  },
  heading:{
    fontSize:38,
    fontWeight: 550,
    textAlign:'center',
    margin: "20px 0px"
  },
  subheading: {
    fontSize:18,
    color: 'grey',
        display:'flex',
        fontWeight: 520,
        margin: "10px 0px",
        [theme.breakpoints.down('sm')]:{
            display:'block'
        }
    },
    details:{
        fontSize:20
    },
    link:{
        color:'inherit',
        textDecoration:'none'
    }
}))
const DetailedView = ({ match }) => {
    const classes = Styles();
    const url = 'https://images.pexels.com/photos/3822335/pexels-photo-3822335.png';
    const history = useHistory();
    const [post,setPost] = useState({});
    useEffect(() => {
        const fetchData = async () => {
            let data = await getPost(match.params.id);
            console.log(data);
            setPost(data);
        }
        fetchData();
    },[match.params.id])
    const deleteBlog= async ()=>{
        await deletePost(post._id);
        history.push('/');
    }
    return(
        <Box className={classes.container}>
            <img src={post.picture || url} alt='banner' className={classes.image}/>
            <Box className={classes.icons}>
              <Link to={`/update/${post._id}`}><Edit className={classes.icon} color='primary'/></Link>
              <Delete onClick={()=> deleteBlog()} className={classes.icon} color='error' cursor='pointer'/>
            </Box>
            <Typography className={classes.heading}>{post.title}</Typography>
            <Box className={classes.subheading}>
              <Typography>Author:&nbsp;
                 <Link to={`/?username=${post.username}`} className={classes.link}>
                    <span style={{fontWeight: 550}}>{post.username}</span>
                </Link>
              </Typography>
              <Typography style={{ marginLeft :'auto'}}>{new
Date(post.createdDate).toDateString()}</Typography>
            </Box>
            <Typography className={classes.details}>{post.description}</Typography>
            <Comments post={post}/>
        </Box>
    )
}
export default DetailedView;
```

5.2.12 UpdateView.jsx [client/scr/component/Post/UpdateView.jsx]

```
import { Box, makeStyles, FormControl, InputBase, Button, TextareaAutosize } from "@material-ui/core";
import { useState, useEffect } from "react";
import { AddCircle } from "@material-ui/icons";
import { getPost, updatePost, uploadFile } from "../../services/api";
import { useHistory } from "react-router-dom";

const Styles = makeStyles((theme) => ({
  container: {
    padding: "0px 100px",
    [theme.breakpoints.down("md")]: {
      padding: 0,
    },
  },
  image: {
    width: "100%",
    height: "50vh",
    objectFit: "cover",
  },
  form: {
    display: "flex",
    flexDirection: "row",
    marginTop: "10px",
  },
  textfield: {
    flex: 1,
    fontSize: 24,
    margin: "0px 25px",
  },
  textarea:{
     marginTop: 10,
     width:'100%',
     border:'none',
     fontSize: 20,
     '&:focus-visible':{
        outline:'none'
     }
  }
}));
const initialValues = {
  title: "",
  description: "",
  picture: "",
  username: "raihanhassan10",
  categories: "All",
  createdDate: new Date(),
};
const UpdateView = ({ match }) => {
  const classes = Styles();
  const history = useHistory();
  const [post,setPost] = useState(initialValues);
  const [file, setFile] = useState('');
  const [image, setImage] = useState('');
  const url = post.picture ? post.picture : "https://images.pexels.com/photos/374631/pexels-photo-
374631.jpeg?auto=compress&cs=tinysrgb&dpr=2&h=650&w=940";

  useEffect (()=> {
    const getImage = async () =>{
      if(file){
        const data = new FormData();
        data.append("name",file.name);
        data.append("file",file);

        let image = await uploadFile(data);
        post.picture = image.data;
        setImage(image.data);
      }
    }
    getImage();
  }, [file,post])

  useEffect(() => {
    const fetchData = async () => {
      let data = await getPost(match.params.id);
      console.log(data);
      setPost(data);
    }
    fetchData();
  },[match.params.id])

  const handleChange = (e) =>{
    setPost({...post,[e.target.name]:e.target.value})
  }
  const updateBlog = async () => {
    await updatePost(match.params.id, post);
    history.push(`/details/${match.params.id}`);
  }

  return (
    <Box className={classes.container}>
      <img src={url} alt="banner" className={classes.image} />
      <FormControl className={classes.form}><label htmlFor='fileInput'>
        <AddCircle fontSize="large" color="error" cursor='pointer' />
        </label>
        <input
          type='file'
          id='fileInput'
          style={{display:'none'}}
          onChange ={(e) => setFile(e.target.files[0])}
        />
        <InputBase placeholder="Title..." name='title' onChange={(e) => handleChange(e)} value={post.title}
className={classes.textfield} />
        <Button onClick={()=> updateBlog()} variant="contained" color="primary">Update</Button>
      </FormControl>
      <TextareaAutosize
         className={classes.textarea}
         rowsMin={5}
         placeholder='Write your story...'
         name='description'
         value={post.description}
         onChange={(e) => handleChange(e)}
      />
    </Box>
  );
};

export default UpdateView;
```

5.2.13 Header.jsx [client/scr/component/Header.jsx]

```
import { AppBar, Toolbar, makeStyles } from '@material-ui/core';
import { Link } from 'react-router-dom';
const useStyle = makeStyles({
    component: {
        background: '#FFFFFF',
        color: 'black'
    },
    container: {
        justifyContent: 'center',
        '&  >*': {
            padding: 20,
            color: 'black',
            textDecoration: 'none'
        }
    }
})
const Header = () => {
    const classes = useStyle();
    return (
        <AppBar className={classes.component}>
            <Toolbar className={classes.container}>
                <Link to='/'>HOME</Link>
                <Link to='/about'>ABOUT</Link>
                <Link to='/contact'>CONTACT</Link>
                {/* <Link to='/'>LOGOUT</Link> */}
            </Toolbar>
        </AppBar>
    )
}
export default Header;
```

5.2.14 index.jsx [client/scr/index.jsx]

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
```

```
ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

5.2.15 index.js [server / index.js]

```
import express from "express";
import cors from 'cors';
import bodyparser from 'body-parser';
import connection from "./database/db.js";
import router from "./routes/route.js";
const app = express();
app.use(cors());
app.use(bodyparser.json({extended:true}));
app.use(bodyparser.urlencoded({extended:true}));
app.use('/', router);
const port = 8000;
app.listen(port,()=> console.log(`Server is running at Port ${port}`));
connection();
```

5.2.16 commentController.js [server / controller / commentController.js]

```
import comment from '../schema/commentSchema.js';
import Comment from '../schema/commentSchema.js';
export const newComment = async (request, response) => {
   try{
      const comment = new Comment(request.body);
      comment.save();
      response.status(200).json('Comment saved successfully');
   }catch(error){
      response.status(500).json(error);
   }
}
export const getComment = async (request, response) => {
  try{
    const comments = await Comment.find({ postId: request.params.id});
    response.status(200).json(comments);
  }catch(error){
    response.status(500).json(error);
  }
}
export const deleteComment = async (request, response) => {
  try{
    const comment = await Comment.findById(request.params.id);
    await comment.delete();
    response.status(200).json(comment);
  }catch(error){
    response.status(500).json(error);
  }
}
```

5.2.17 imageController.js [server / controller / imageController.js]

```
import Grid from "gridfs-stream";
import mongoose from "mongoose";
const url = "http://localhost:8000";
let gfs;
const conn = mongoose.connection;
conn.once("open", () => {
  gfs = Grid(conn.db, mongoose.mongo);
  gfs.collection("fs");
});
export const uploadImage = (request, response) => {
  try {
    if (!request.file) return response.status(404).json("File not found");
    const imageUrl = `${url}/file/${request.file.filename}`;
    response.status(200).json(imageUrl);
  } catch (error) {
    response.status(404).json(error);
  }
}
```

```
export const getImage = async (request, response) => {
  try {
    const file = await gfs.files.findOne({ filename: request.params.filename });
    let readStream = gfs.createReadStream(file.filename);
    readStream.pipe(response);
  } catch (error) {
    response.status(500).json("Failed to fetch the image");
  }
}
```

5.2.18 postController.js [server / controller / postController.js]

```
import Post from "../schema/postSchema.js";
export const createPost = async (request,response) => {
   console.log(request.body);
   try{
      const post = await new Post(request.body);
      post.save();
      response.status(200).json('blog saved successfully');
   }catch(error){
      response.status(500).json(error);
   }
}
export const getAllPosts = async (request,response) => {
   let posts;
   let username = request.query.username;
   let category = request.query.category;
   try{
      if(username){
         posts = await Post.find({ username: username});
      }
      else if(category){
         posts = await Post.find({ categories: category});
      }
      else{
         posts = await Post.find({});
```

```
    }
    response.status(200).json(posts);
  }catch (error) {
    response.status(500).json(error);
  }
}
export const getPost = async (request, response) => {
  try{
    let post = await Post.findById(request.params.id);
    response.status(200).json(post);
  }catch(error){
    response.status(500).json(error);
  }
}
export const updatePost = async (request, response) => {
  try{
    await Post.findByIdAndUpdate(request.params.id,{$set: request.body});
    response.status(200).response('Blog Updated');
  }catch(error){
    response.status(500).json(error);
  }
}
export const deletePost = async (request, response) => {
  try{
    let post = await Post.findById(request.params.id);
    await post.delete();
    response.status(200).response('Blog deleted');
  }catch(error){
    response.status(500).json(error);
  }
}
```

5.2.19 db.js [server / database / db.js**]**

```
import mongoose from 'mongoose';
```

```
const connection = async () => {
  try{

    const url
='mongodb+srv://raihanhassan10:ANUBHAVLALROX@cluster0.x6htlxy.mongodb.net/?retryWrites=true
&w=majority';

    await mongoose.connect(url,{useNewUrlParser: true, useUnifiedTopology:true});
    console.log('Database connection established');
  }catch(error){
    console.log('Error while connecting to database',error);
  }
}

export default connection;
```

5.2.20 route.js [server / routes / route.js]

```
import express, { Router } from 'express';
import { createPost, getAllPosts, getPost, updatePost, deletePost } from "../controller/postController.js";
import { uploadImage, getImage } from '../controller/imageController.js';
import upload from '../utilities/upload.js'
import { newComment,getComment,deleteComment } from '../controller/commentController.js';
const router = express.Router();
router.post('/create', createPost);
router.get('/posts', getAllPosts);
router.get('/post/:id', getPost);
router.post('/update/:id', updatePost);
router.delete('/delete/:id', deletePost);
router.post('/file/upload', upload.single('file'), uploadImage);
router.get('/file/:filename', getImage);
router.post('/comment/new', newComment);
router.get('/comments/:id', getComment);
router.delete('/comment/delete/:id', deleteComment);
export default router;
```

5.2.21 commentSchema.js [server / schema / commentSchema.js]

```
import mongoose from 'mongoose';
```

```
const commentSchema =  mongoose.Schema({
  name:{
    type: String,
    require:true
  },
  postId:{
    type: String,
    require:true
  },
  date:{
    type: String,
    require:true
  },
  comments:{
    type: String,
    require:true
  }
})
const comment = mongoose.model('comment',commentSchema);
export default comment;
```

5.2.22 postSchema.js [server / schema / postSchema.js]

```
import mongoose from 'mongoose';
const postSchema = mongoose.Schema({
  title:{
    type:String,
    required:true,
    unique:true
  },
  description:{
    type:String,
    required:true,
  },
  picture:{
    type:String,
    required:false,
  },
  username:{
    type:String,
    required:true,
  },
  categories:{
    type:String,
    required:true,
  },
  createdDate:{
    type:Date
  },
})
const post = mongoose.model('post',postSchema);
export default post;
```

5.2.23 upload.js [server / utilities / upload.js]

```
import multer from 'multer';
import { GridFsStorage } from 'multer-gridfs-storage';
const storage = new GridFsStorage({

url:'mongodb+srv://raihanhassan10:ANUBHAVLALROX@cluster0.x6htlxy.mongodb.net/?retryWrites=tru
e&w=majority',
  options: { useNewUrlParser: true, useUnifiedTopology:true },
  file: (request,file) =>{
    const match = ["image/png","image/jpg"];


    if(match.indexOf(file.mimeType) === -1)
    {
      return `${Date.now()}-blog-${file.originalname}`;
    }
    else{
      return {
        bucketName: 'photos',
        filename: `${Date.now()}-blog-${file.originalname}`
      }
    }
  }
})

export default multer ({storage});
```

6.TESTING AND REPORT

6.1 INTRODUCTION TO SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

6.2 TYPES OF TESTING: -

6.2.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.2.3 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

Valid Input: identified classes of valid input must be accepted.

Invalid Input: identified classes of invalid input must be rejected.

Functions: identified functions must be exercised.

Output: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

6.2.4 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

6.2.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

6.2.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

6.2.7 Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

• All field entries must work properly.

• Pages must be activated from the identified link.

• The entry screen, messages and responses must not be delayed.

Features to be tested

• Verify that the entries are of the correct format

• No duplicate entries should be allowed

• All links should take the user to the correct page.

Integration Testing:

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g., components in a software system or – one step up – software applications at the company level – interact without error.

Test Results:

All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing:

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results:

All the test cases mentioned above passed successfully. No defects encountered.

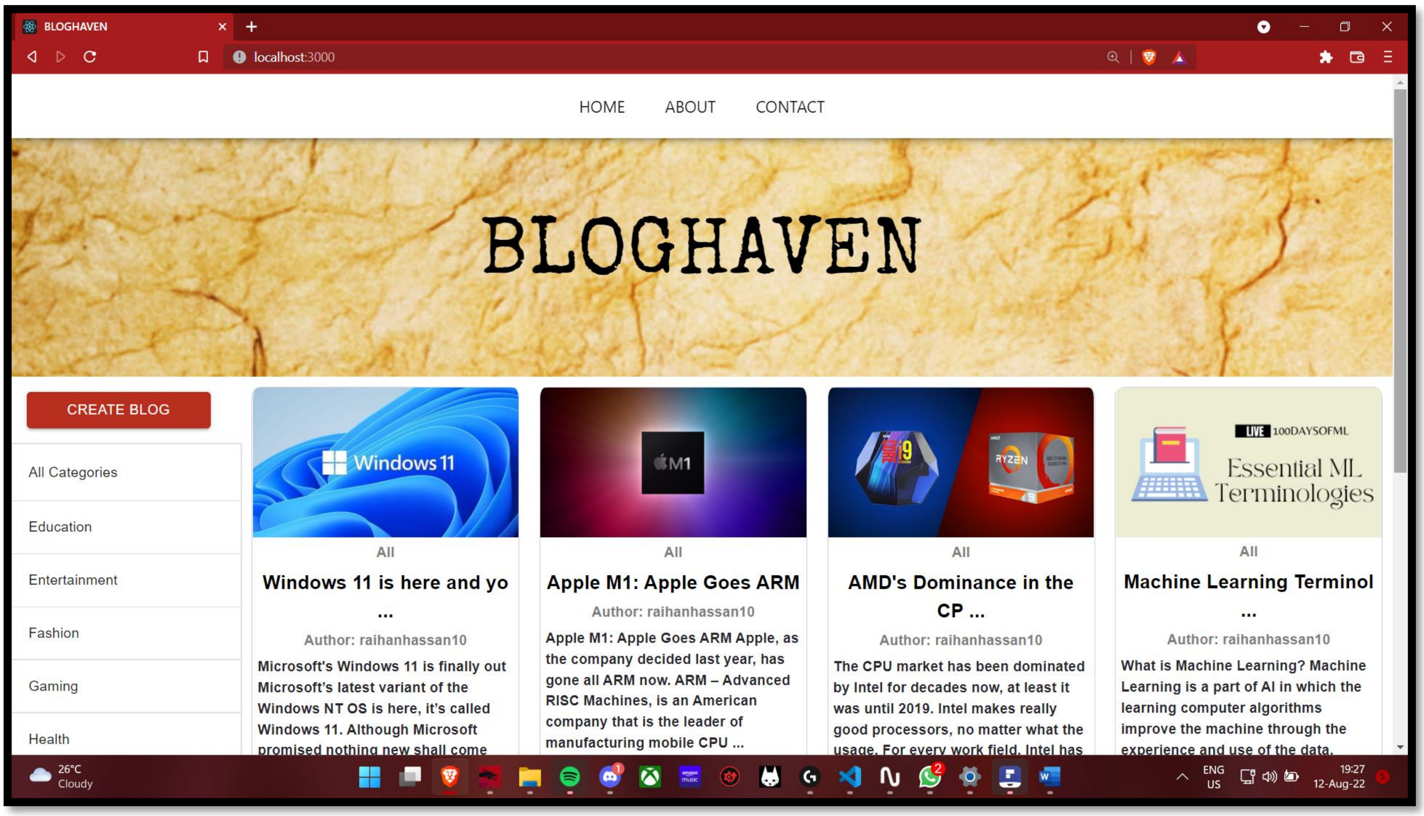
Below are the test cases for the Job Search Portal web application.

TEST MODULE	TEXT CASE	EXPECTED RESULT	TEST RESULT
USER	Creates a blog with valid title and description.	The blog gets successfully saved in the database and gets displayed on the home page.	PASS
USER	Creates a blog with invalid title or description.	The application gets redirected to the home page and the data gets rejected.	PASS

USER	Tries to delete a blog from the pool of posts.	The blog gets deleted from the array of posts and from the database.	PASS
USER	Tries to update the details of the blog like title, description or the image.	The blog gets updated in the database and gets reflected back in the application.	PASS

7. SAMPLE SCREENSHOTS

HOME PAGE



All

Windows 11 is here and yo ...

Author: raihanhassan10

Microsoft's Windows 11 is finally out Microsoft's latest variant of the Windows NT OS is here, it's called Windows 11. Although Microsoft promised nothing new shall come

All

Apple M1: Apple Goes ARM

Author: raihanhassan10

Apple M1: Apple Goes ARM Apple, as the company decided last year, has gone all ARM now. ARM – Advanced RISC Machines, is an American company that is the leader of manufacturing mobile CPU ...

All

AMD's Dominance in the CP ...

Author: raihanhassan10

The CPU market has been dominated by Intel for decades now, at least it was until 2019. Intel makes really good processors, no matter what the usage. For every work field, Intel has

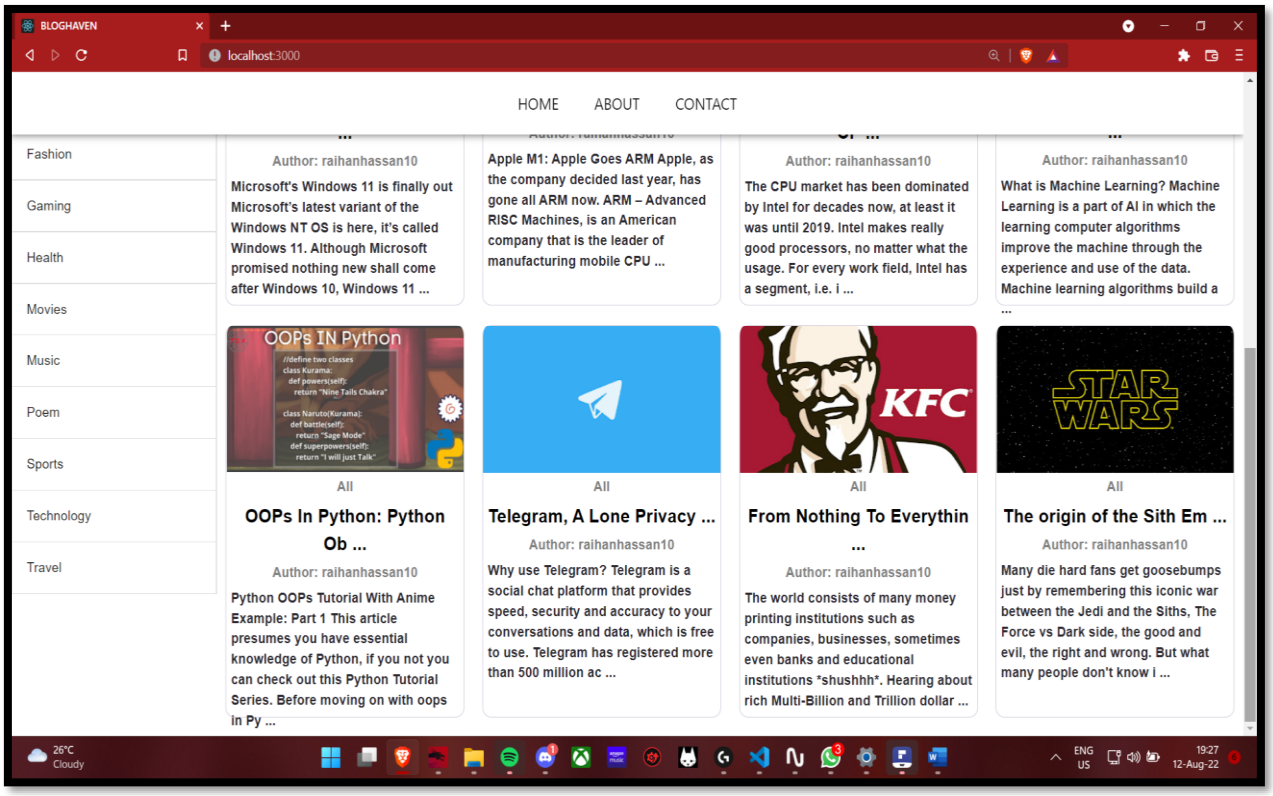
All

Machine Learning Terminol ...

Author: raihanhassan10

What is Machine Learning? Machine Learning is a part of AI in which the learning computer algorithms improve the machine through the experience and use of the data.

Home page displays *the entire content* which is stored in the database



...
Author: raihanhassan10
Microsoft's Windows 11 is finally out Microsoft's latest variant of the Windows NT OS is here, it's called Windows 11. Although Microsoft promised nothing new shall come after Windows 10, Windows 11 ...

Apple M1: Apple Goes ARM Apple, as the company decided last year, has gone all ARM now. ARM – Advanced RISC Machines, is an American company that is the leader of manufacturing mobile CPU ...

Author: raihanhassan10
The CPU market has been dominated by Intel for decades now, at least it was until 2019. Intel makes really good processors, no matter what the usage. For every work field, Intel has a segment, i.e. i ...

Author: raihanhassan10
What is Machine Learning? Machine Learning is a part of AI in which the learning computer algorithms improve the machine through the experience and use of the data. Machine learning algorithms build a ...

All
OOPs In Python: Python Ob ...
Author: raihanhassan10
Python OOPs Tutorial With Anime Example: Part 1 This article presumes you have essential knowledge of Python, if you not you can check out this Python Tutorial Series. Before moving on with oops in Py ...

All
Telegram, A Lone Privacy ...
Author: raihanhassan10
Why use Telegram? Telegram is a social chat platform that provides speed, security and accuracy to your conversations and data, which is free to use. Telegram has registered more than 500 million ac ...

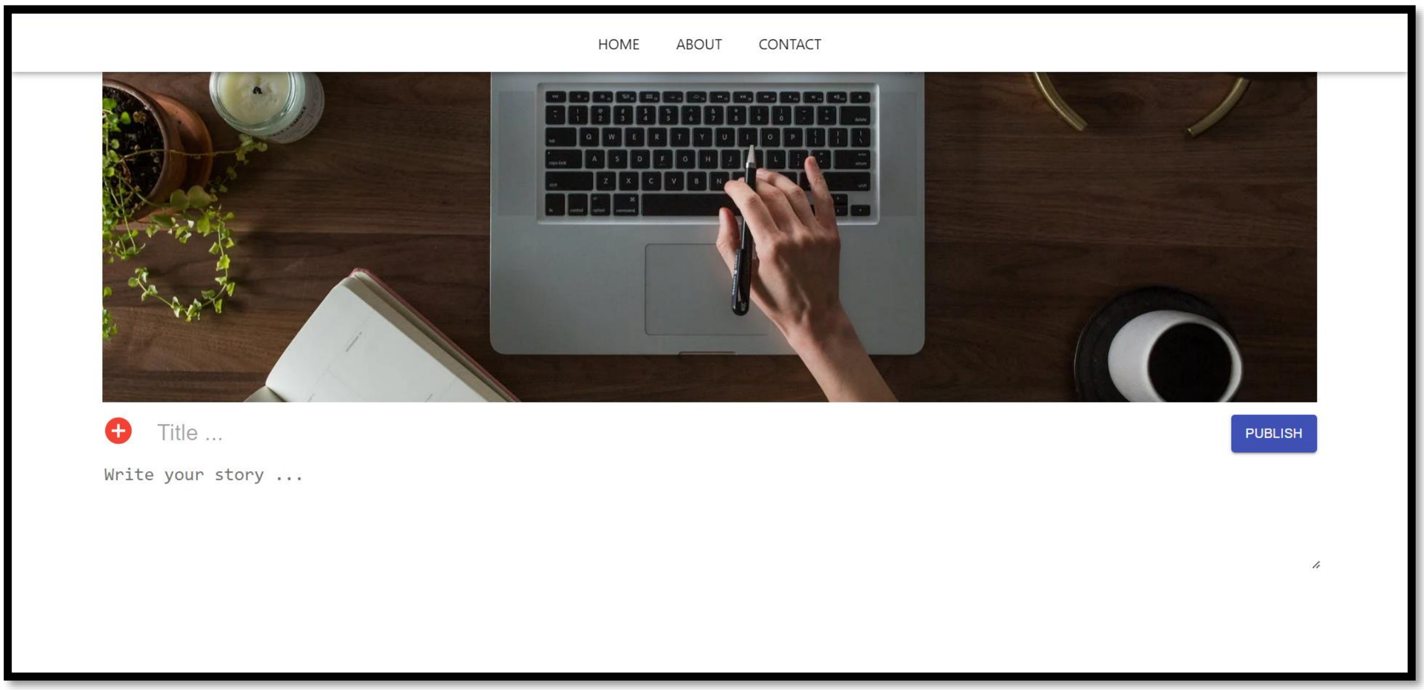
All
From Nothing To Everythin ...
Author: raihanhassan10
The world consists of many money printing institutions such as companies, businesses, sometimes even banks and educational institutions *shushhh*. Hearing about rich Multi-Billion and Trillion dollar ...

All
The origin of the Sith Em ...
Author: raihanhassan10
Many die hard fans get goosebumps just by remembering this iconic war between the Jedi and the Siths, The Force vs Dark side, the good and evil, the right and wrong. But what many people don't know i ...

Home page display the content filtered by *username*

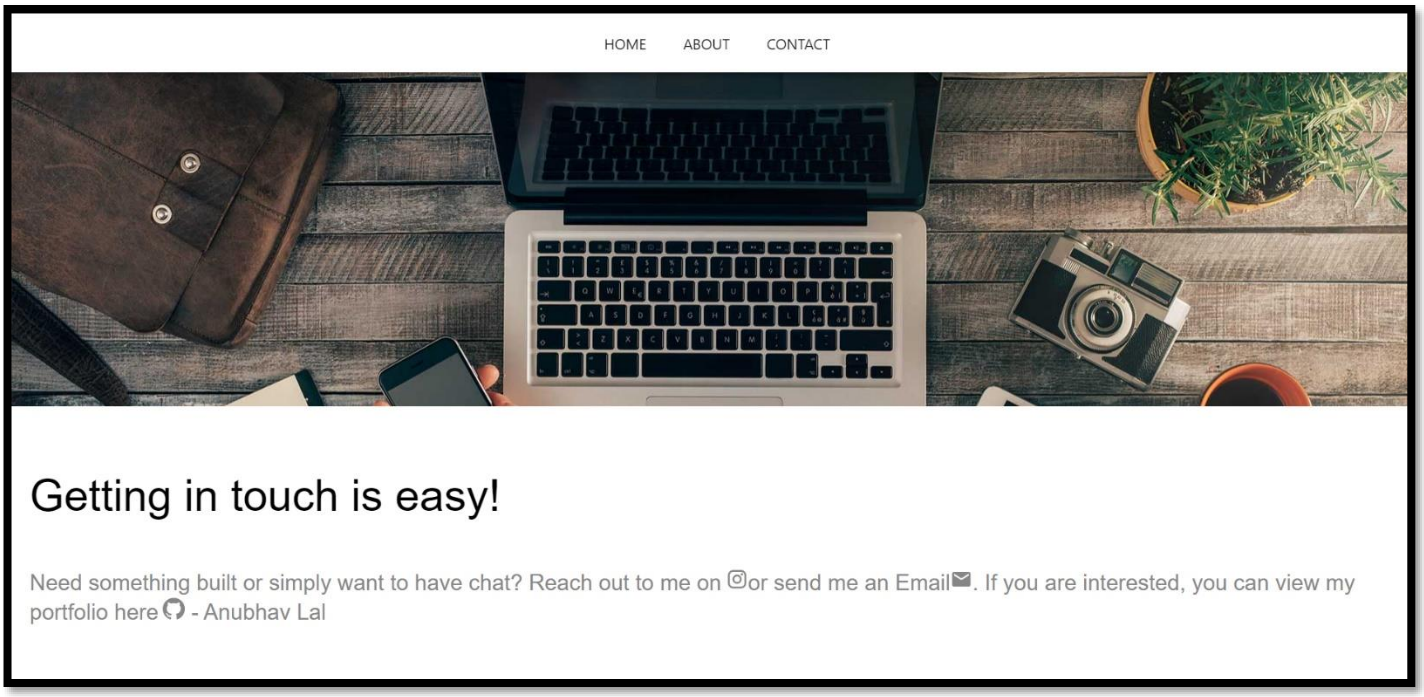
Home page displaying the content filtered by *category*

CREATE VIEW



This is the page where the user creates the blog with *title, description and an image*

CONTACT VIEW

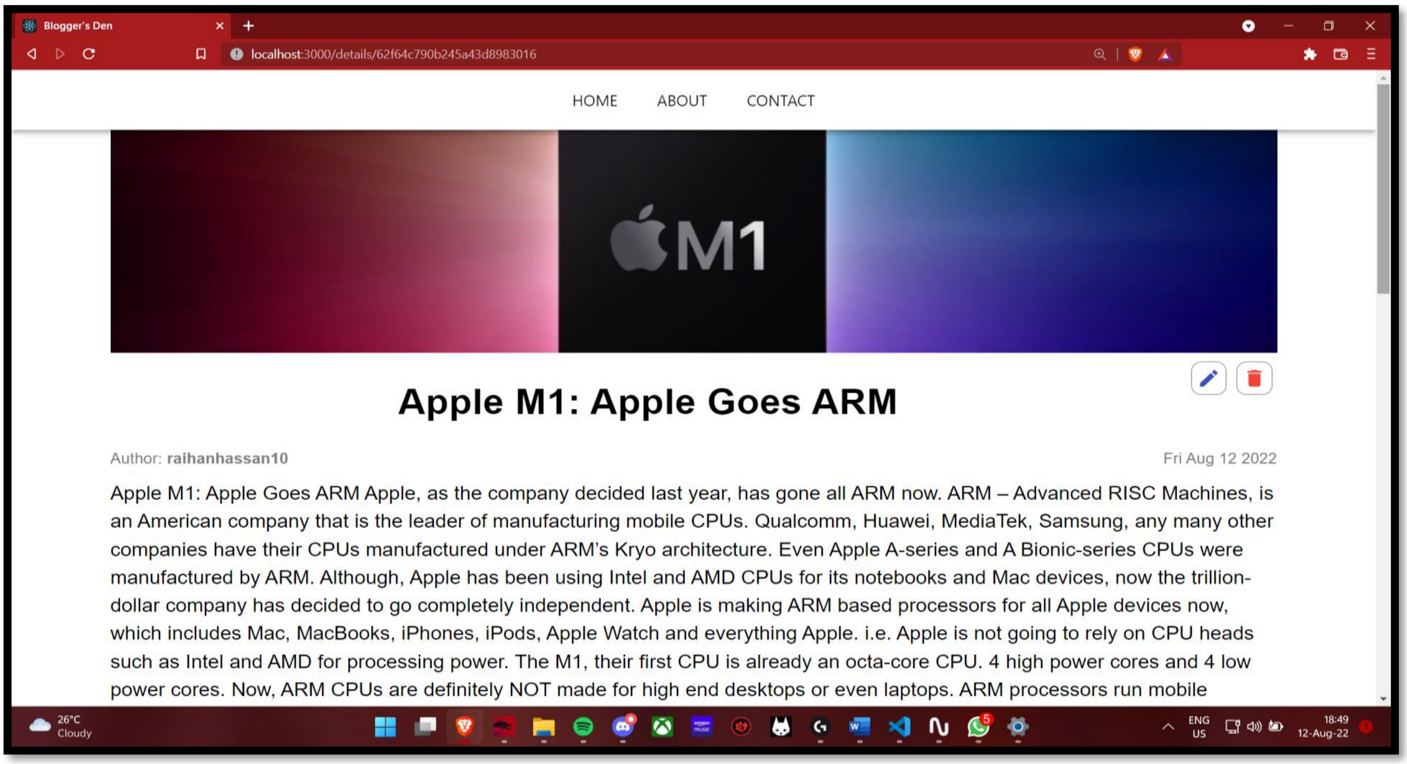


Getting in touch is easy!

Need something built or simply want to have chat? Reach out to me on or send me an Email. If you are interested, you can view my portfolio here - Anubhav Lal

Contact view displays the contact information of the *developer/designer*

DETAILED VIEW



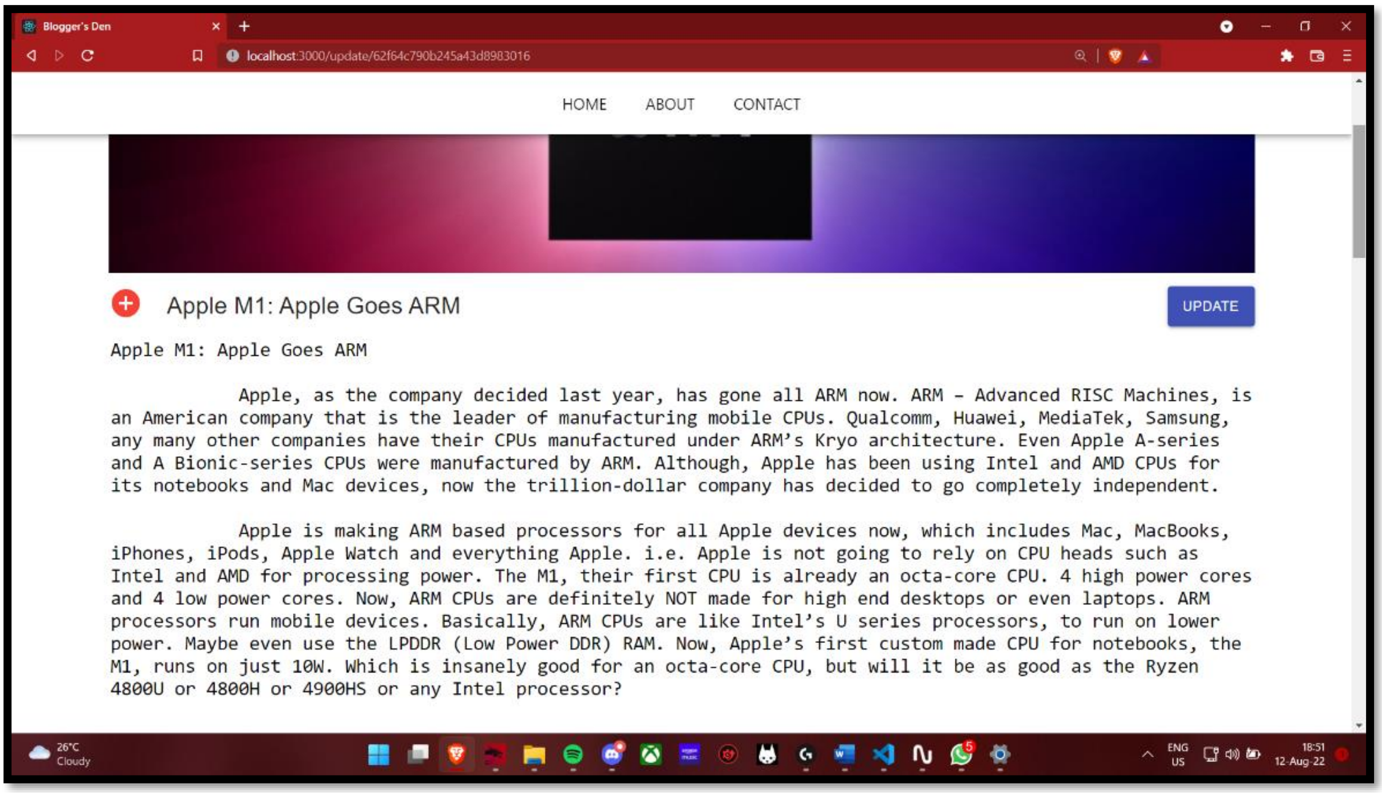
Apple M1: Apple Goes ARM

Author: raihanhassan10 Fri Aug 12 2022

Apple M1: Apple Goes ARM Apple, as the company decided last year, has gone all ARM now. ARM – Advanced RISC Machines, is an American company that is the leader of manufacturing mobile CPUs. Qualcomm, Huawei, MediaTek, Samsung, any many other companies have their CPUs manufactured under ARM's Kryo architecture. Even Apple A-series and A Bionic-series CPUs were manufactured by ARM. Although, Apple has been using Intel and AMD CPUs for its notebooks and Mac devices, now the trillion-dollar company has decided to go completely independent. Apple is making ARM based processors for all Apple devices now, which includes Mac, MacBooks, iPhones, iPods, Apple Watch and everything Apple. i.e. Apple is not going to rely on CPU heads such as Intel and AMD for processing power. The M1, their first CPU is already an octa-core CPU. 4 high power cores and 4 low power cores. Now, ARM CPUs are definitely NOT made for high end desktops or even laptops. ARM processors run mobile

Detailed view displays the title, description and the image which is stored in the database

UPDATE VIEW



Apple M1: Apple Goes ARM

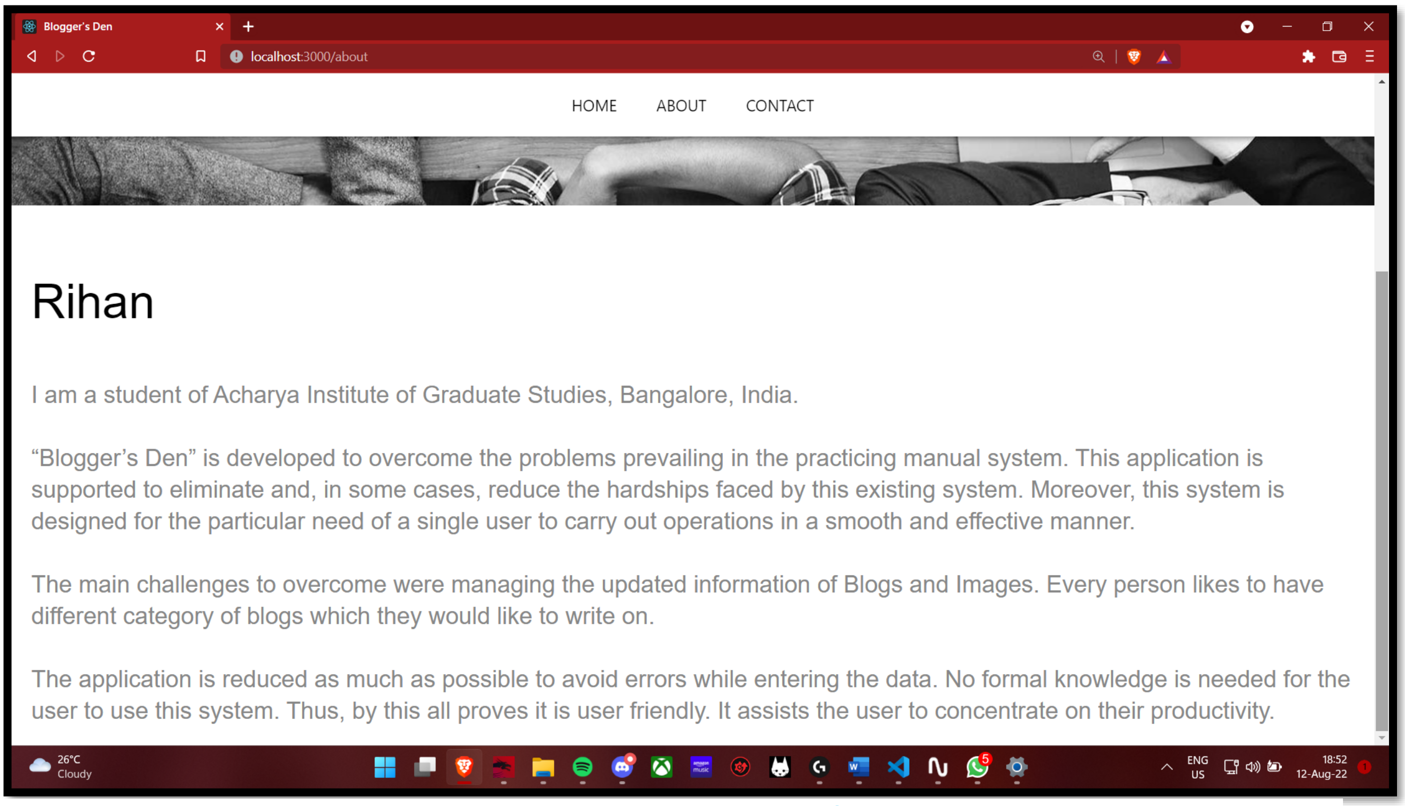
Apple M1: Apple Goes ARM

Apple, as the company decided last year, has gone all ARM now. ARM – Advanced RISC Machines, is an American company that is the leader of manufacturing mobile CPUs. Qualcomm, Huawei, MediaTek, Samsung, any many other companies have their CPUs manufactured under ARM's Kryo architecture. Even Apple A-series and A Bionic-series CPUs were manufactured by ARM. Although, Apple has been using Intel and AMD CPUs for its notebooks and Mac devices, now the trillion-dollar company has decided to go completely independent.

Apple is making ARM based processors for all Apple devices now, which includes Mac, MacBooks, iPhones, iPods, Apple Watch and everything Apple. i.e. Apple is not going to rely on CPU heads such as Intel and AMD for processing power. The M1, their first CPU is already an octa-core CPU. 4 high power cores and 4 low power cores. Now, ARM CPUs are definitely NOT made for high end desktops or even laptops. ARM processors run mobile devices. Basically, ARM CPUs are like Intel's U series processors, to run on lower power. Maybe even use the LPDDR (Low Power DDR) RAM. Now, Apple's first custom made CPU for notebooks, the M1, runs on just 10W. Which is insanely good for an octa-core CPU, but will it be as good as the Ryzen 4800U or 4800H or 4900HS or any Intel processor?

Update view allows the user to modify the content of the post

ABOUT VIEW



Rihan

I am a student of Acharya Institute of Graduate Studies, Bangalore, India.

"Blogger's Den" is developed to overcome the problems prevailing in the practicing manual system. This application is supported to eliminate and, in some cases, reduce the hardships faced by this existing system. Moreover, this system is designed for the particular need of a single user to carry out operations in a smooth and effective manner.

The main challenges to overcome were managing the updated information of Blogs and Images. Every person likes to have different category of blogs which they would like to write on.

The application is reduced as much as possible to avoid errors while entering the data. No formal knowledge is needed for the user to use this system. Thus, by this all proves it is user friendly. It assists the user to concentrate on their productivity.

About view displays the *description of the project*

8. CONCLUSION

Our project is only a humble venture to satisfy the needs to manage their project work. Several user-friendly coding has also adopted. This package shall prove to be a powerful package in satisfying all the requirements of the college. The objective of software planning is to provide a framework that enables the user to create a blog with minimal efforts.

At the end it is concluded that we have made effort on the following points: -

- A description of the background and context of the project and its relation to work already done in the area.
- We mentioned the technologies used in order to make this project.
- We understood the problems faced the by the user and tried to developed an application to eradicate the problem.
- We included features and operations in detail, including screen layouts.

Starting from requirements elicitation to design, construction, implementation and testing, I have gained a very good experience working with various technologies at every phase. Development of this project boosted my confidence in web development.

9. FUTURE ENHANCHEMENTS

- ✓ We can give more advance software for blogging application including more facilities.
- ✓ We will host platform on online servers to make it accessible worldwide.
- ✓ Integrated multiple load balancer to distribute the loads of the system.
- ✓ Create the master and slave database structure to reduce the overload of the database queries.
- ✓ Implement the backup mechanism for taking backup of codebase and database on regular basis on different servers.

The above-mentioned points are the enhancement which can be done to increase the applicable and usage of this project. Here we can maintain the records of blogs and comment. Also, as it can be seen now-a-days the players are versatile, i.e., so there is a scope for introducing a method to maintain all the blogs, comment and image.

10. BIBLOGRAPHY

https://www.mongodb.com/mern-stack?fbclid=IwAR19AW5xLR45sMUccHB1mjbZLbAq8u8ePnmEI6aAYlM1H2vEtQtpAwrSYSU

https://www.mongodb.com/mern-stack?fbclid=IwAR19AW5xLR45sMUccHB1mjbZLbAq8u8ePnmEI6aAYlM1H2vEtQtpAwrSYSU

https://docs.mongodb.com/manual/introduction/

https://mongoosejs.com/docs/guide.html

https://en.wikipedia.org/wiki/Express.js#:~:text=js%2C%20or%20simply%20Express%2C%20is,software%20under%20the%20MIT%20License.&text=js.,many%20features%20available%20as%20plugins

https://expressjs.com/en/guide/using-middleware.html

https://blog.npmjs.org/post/615388323067854848/so-long-and-thanks-for-all-the-packages.html#:~:text=In%20June%202019%2C%20the%20npm,number%20has%20crossed%201.3%20million

https://reactjs.org/docs/getting-started.html

https://reactjs.org/docs/faq-internals.html

https://reactjs.org/docs/components-and-props.html

https://reactjs.org/docs/state-and-lifecycle.html

https://www.mongodb.com/mern-stack?fbclid=IwAR19AW5xLR45sMUccHB1mjbZLbAq8u8ePnmEI6aAYlM1H2vEtQtpAwrSYSU

https://codersontrang.wordpress.com/2017/10/09/gioi-thieu-ve-nodejs/

https://blog.npmjs.org/post/615388323067854848/so-long-and-thanks-for-all-the-packages.html#:~:text=In%20June%202019%2C%20the%20npm,number%20has%20crossed%201.3%20million