



EXTRA SERIOUS COMPANY LTD

INCIDENT MANAGEMENT SYSTEM

TABLE OF CONTENTS

PROJECT DESCRIPTION	3
Workflow	3-4
Wireframes	5
Login screen	5
Dashboard	5
Incidence details	6
Screenshots	7
Database users' collection	7
Incidents collection	7
Incidents loaded on the system	8
Incident details window display	8
Incident creation window	9
User account creation window	9
Dashboard	10
Code snippets	11
Incidents component	12
Incidents routes	13
Backend server	15
Future functionalities	16
Project URL	16

PROJECT DESCRIPTION

Incident management system will used in managing customer incidences. The system will have users. Users will be required to register before using the system. Users will be able to add incidences, give comments on the incidences and change the incidence status.

The system will have a dashboard for managing incidences. By default, closed incidences will not be displayed. However, there will be an option for viewing all incidences including closed incidences.

WORKFLOW

1. User Management and site security:
 - a. User Registration must be included. A form will allow the user to enter profile information (username, password, email address, user type) which will be stored in a MongoDB database table.
 - b. The user will be able to Login, Logout and modify his or her.
 - c. Site security will prevent non-registered users from creating incident records (tickets), changing ticket status, posting a comment or seeing the incident log.
2. Incident Dashboard:
 - a. After a user is registered and logged in, he or she can view an Incident Dashboard that will display all open Incidents (tickets) in a clickable list format
 - b. The Dashboard will include an option that allows the user to create a new incident (ticket
 - c. Incidents that are closed will be initially hidden but an option on the dashboard will allow the user to view ALL incidents
3. Create an Incident Record:
 - a. When a new incident is created a form will be displayed that will require the user to select and/or complete several fields including: Incident Description, Incident priority, Customer information and Incident Narrative.

- b. The new incident record will be stored in numerical order and the incident **record number** will be generated based on the incident date (e.g. 130418-0000001). This is typically the number provided to the customer as a reference
- c. Each Incident Record (Ticket) will include an **Incident Narrative** that will be **timestamped** with every status change or Incident modification. This will provide detailed incident information as well as an audit
- d. Each Incident Record (Ticket) will have a **Status Field** associated with it. Initially the Status field will be set to NEW.

4. Incident Management

- a. A registered user can change the **Status Field** of an Incident by first selecting it on the Dashboard and then selecting the appropriate Status (e.g. In Progress, Dispatched, Closed, etc.). The user must then enter a comment in the **Incident Narrative**
- b. Once the status of an Incident Record is set to CLOSED it will not accept any further modifications.
- c. Certain fields will appear **greyed-out** and will not be modifiable (e.g. Incident Record Number, Customer Name, Incident Duration, etc.)
- d. Every active Incident Record will include **an Incident Resolution Field** which must be filled out before a ticket can be officially closed

WIREFRAMES

LOGIN SCREEN

INCIDENT MANAGEMENT SYSTEM

Email address

Password

LOGIN

DASHBOARD

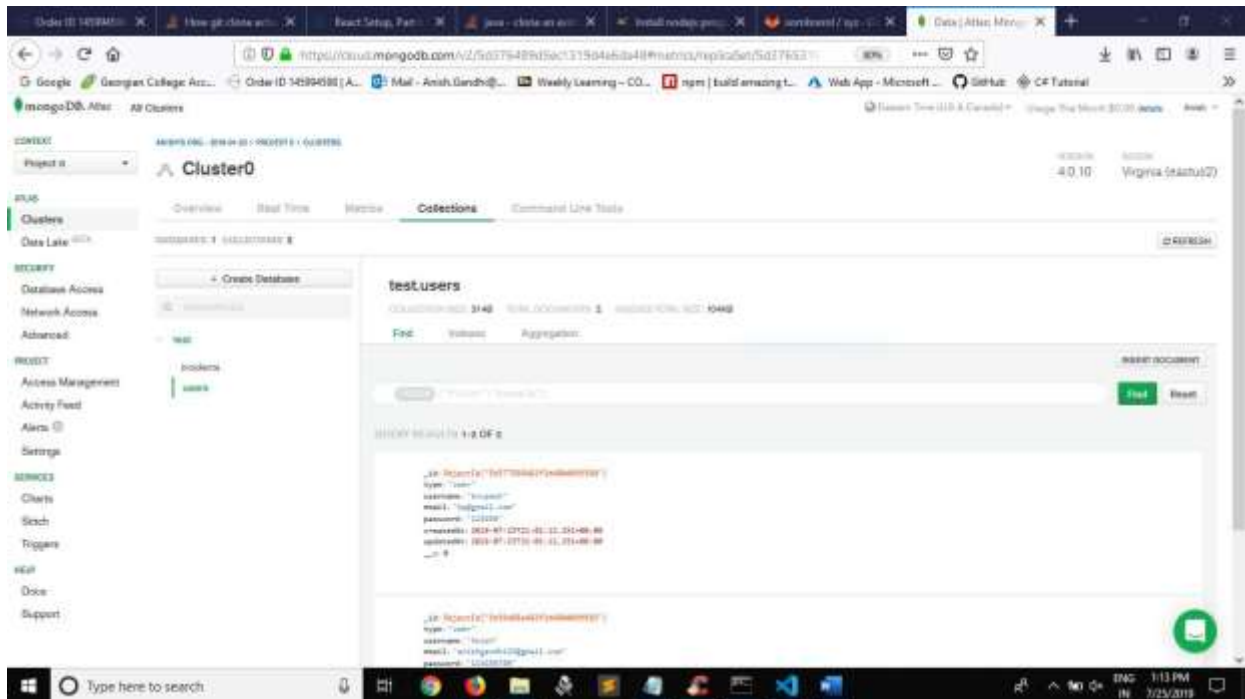
Incidence Management SYstem Dashboard				
Dashboard	Latest Incidences			
Incidences				
Users				

INCIDENCE DETAILS

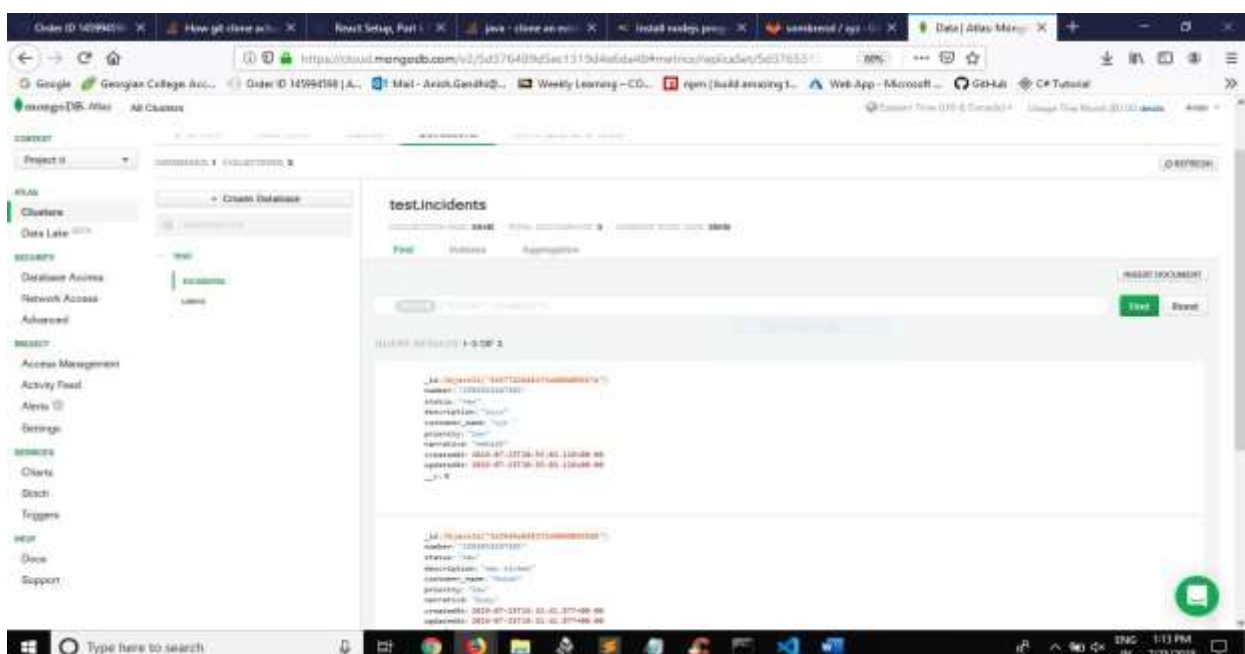
Incidence Management SYstem Dashboard	
Dashboard	Incidence details
Incidences	Incidence No
Users	Description
	Customer
	Status
	LOGS

SCREENSHOTS

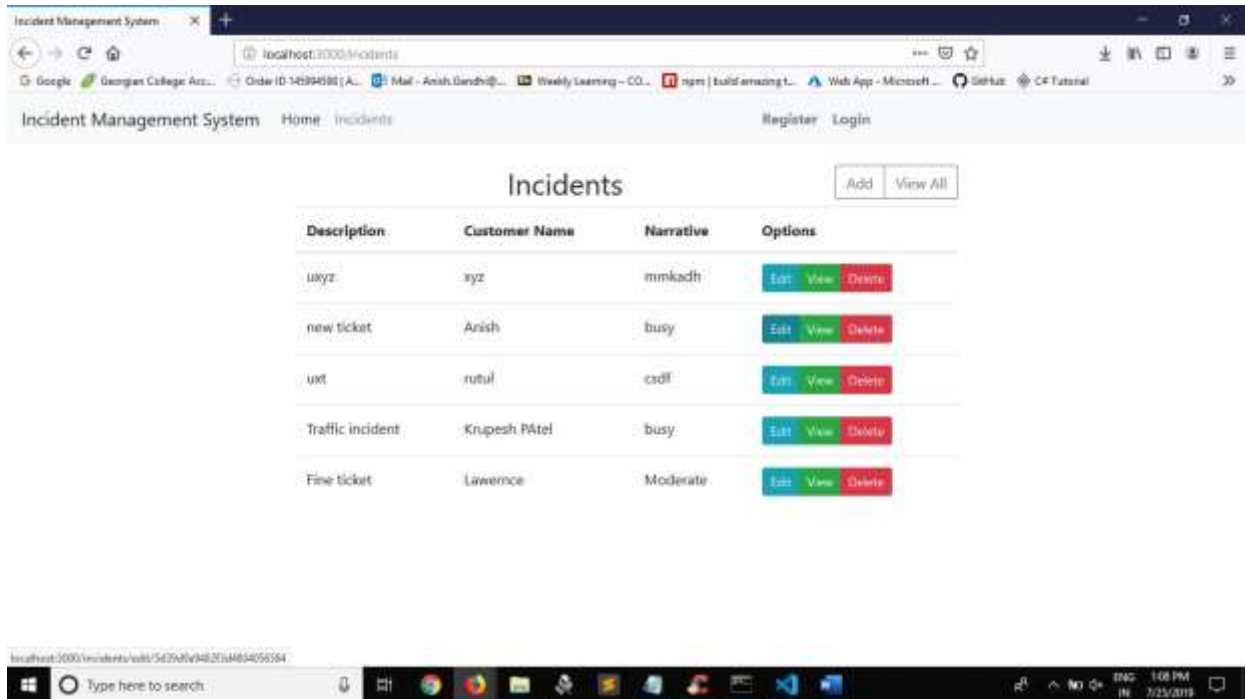
DATABASE USERS' COLLECTION



INCIDENTS COLLECTION



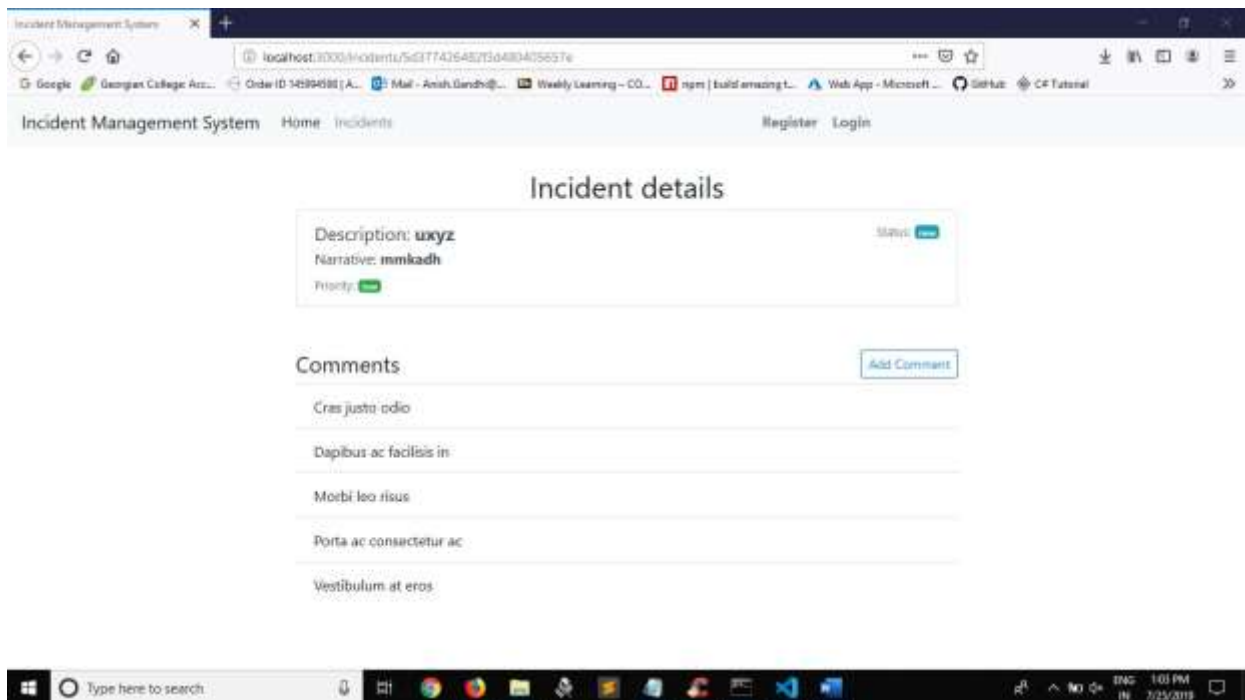
INCIDENTS LOADED ON THE SYSTEM



The screenshot displays the 'Incidents' page of the Incident Management System. The page has a navigation bar with 'Home' and 'Incidents' links, and 'Register' and 'Login' buttons. Below the navigation bar, there is a table titled 'Incidents' with columns: 'Description', 'Customer Name', 'Narrative', and 'Options'. The table contains six rows of incident data. Each row has 'Edit', 'View', and 'Delete' buttons in the 'Options' column. The browser's address bar shows the URL 'localhost:3000/incidents'.

Description	Customer Name	Narrative	Options
wxyz	xyz	mmkadh	Edit View Delete
new ticket	Anish	busy	Edit View Delete
uxi	rutul	cadf	Edit View Delete
Traffic incident	Krupesh Patel	busy	Edit View Delete
Fine ticket	Lawrence	Moderate	Edit View Delete

INCIDENT DETAILS WINDOW DISPLAY



The screenshot displays the 'Incident details' page of the Incident Management System. The page has a navigation bar with 'Home' and 'Incidents' links, and 'Register' and 'Login' buttons. Below the navigation bar, there is a form titled 'Incident details' with fields for 'Description', 'Narrative', and 'Priority'. The 'Description' field contains the text 'wxyz', 'Narrative' contains 'mmkadh', and 'Priority' contains 'low'. There is a 'Status' dropdown menu set to 'Open'. Below the form, there is a 'Comments' section with a list of placeholder text: 'Cras justo odio', 'Dapibus ac facilisis in', 'Morbi leo risus', 'Porta ac consectetur ac', and 'Vestibulum at eros'. There is an 'Add Comment' button next to the comments list. The browser's address bar shows the URL 'localhost:3000/incidents/5d377435482f3d483d405657e'.

Incident details

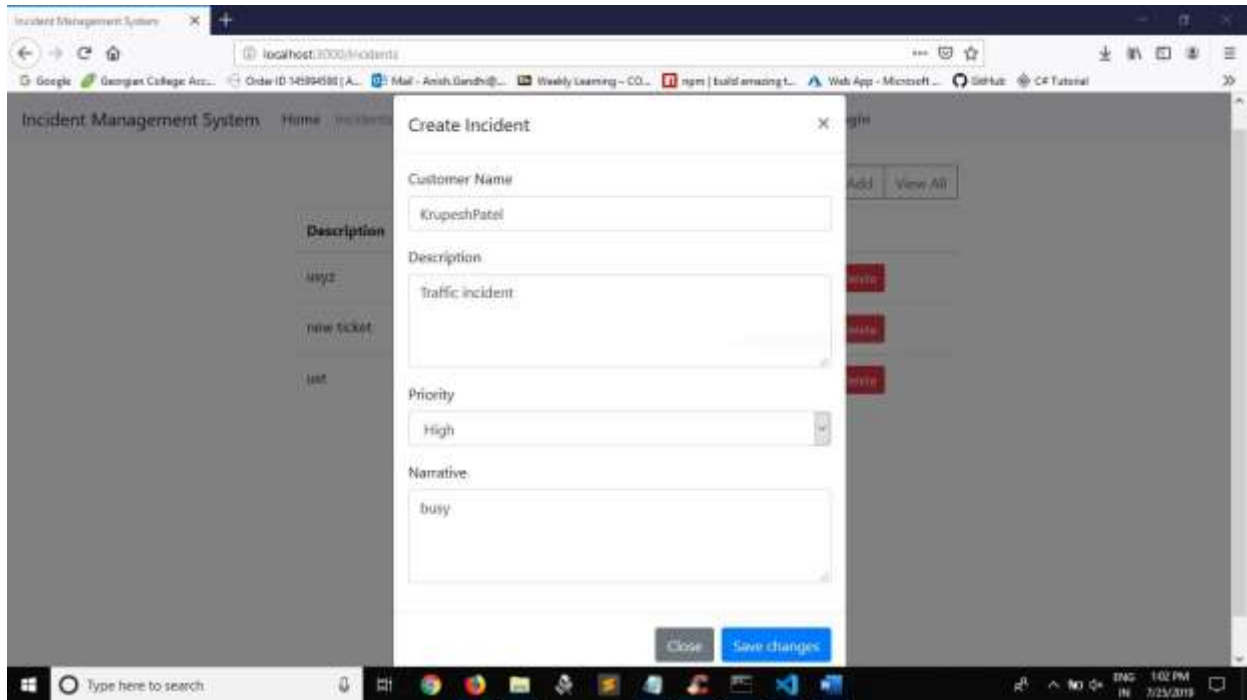
Description: wxyz
Narrative: mmkadh
Priority: low
Status: Open

Comments

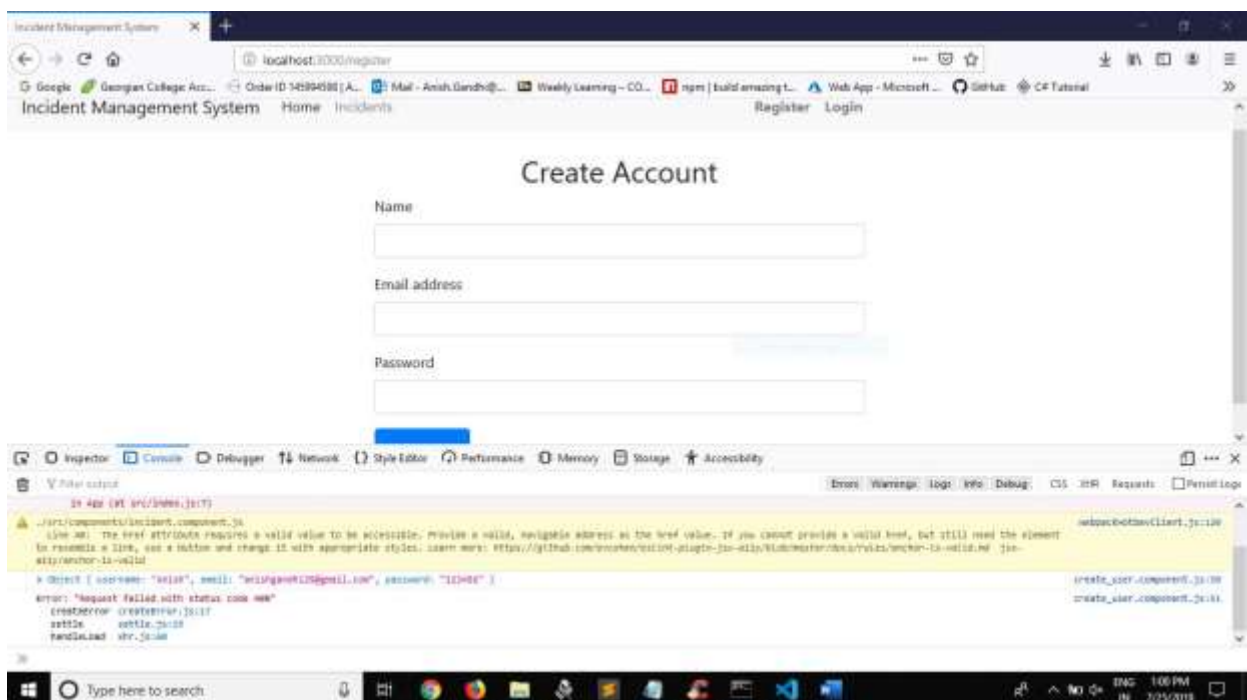
Cras justo odio
Dapibus ac facilisis in
Morbi leo risus
Porta ac consectetur ac
Vestibulum at eros

Add Comment

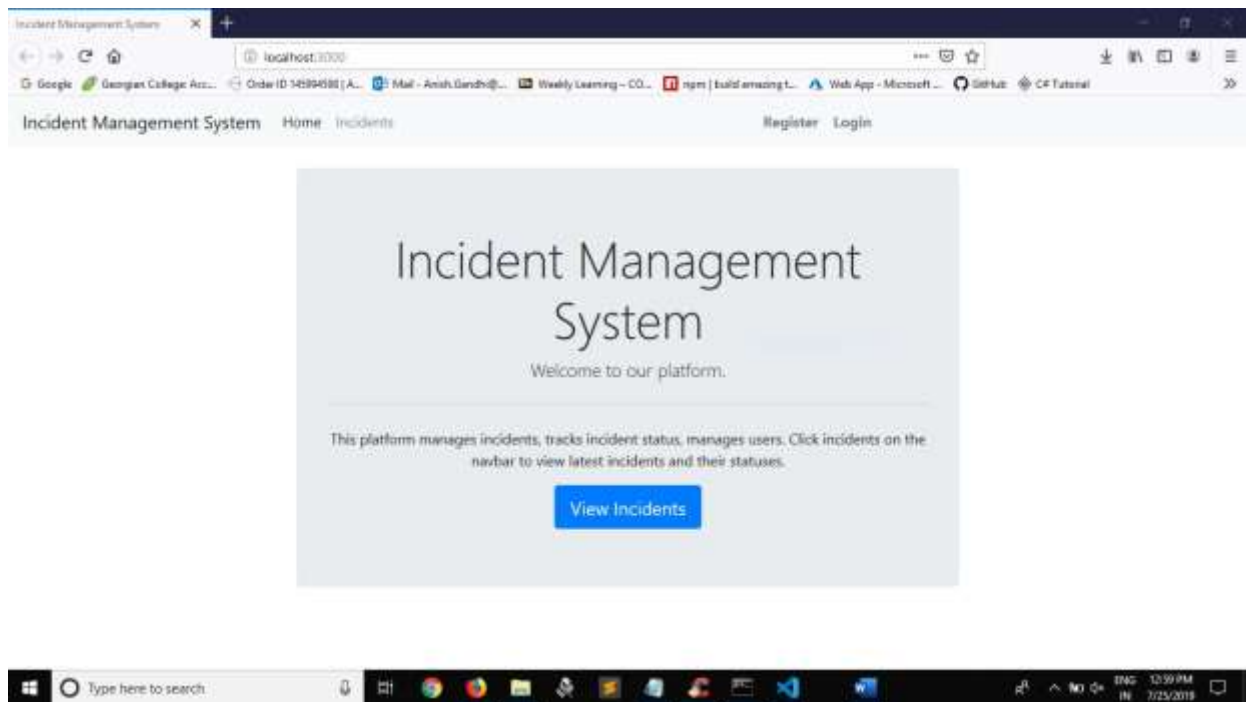
INCIDENT CREATION WINDOW



USER ACCOUNT CREATION WINDOW



DASHBOARD



CODE SNIPPETS

INCIDENTS COMPONENT

```
import React,{Component} from 'react'; import
axios from "axios";
// import {Link} from 'react-router-dom';

export default class Users extends Component {
  constructor(props){    super(props);    this.state
= {          description:",          narrative:",
priority:",          status:",          customer:",
comments:[]
    };

  }

  componentDidMount() {
    console.log(this.props.match.params.id);
    axios.get("http://localhost:5000/incidents/"+this.props.match.params.id)
      .then(res=>{    this.setState({
        description:res.data.description,
narrative:res.data.narrative,          priority:res.data.priority,
status:res.data.status,
        customer:res.data.customer_name
      })
    })
    .catch(err=>console.log(err));
  }
}
```

```

render() {      return
(
    <div className="row">
        <div className="col-sm-8 m-auto">
            <h1 className="h2 text-center">Incident details</h1>
            <div className="list-group">
                <a href="#" className="list-group-item list-group-itemaction">
                    <div className="d-flex w-auto justify-content-between">
                        <h5 className="mb-1">Description:
<b>{this.state.description}</b></h5><br/>
                        <small className="text-muted">Status: <span className="badge badge-
info"> {this.state.status}</span></small>
                    </div>
                    <p className="mb-
1">Narrative:
<b>{this.state.narrative}</b></p>
                        <small className="text-muted">Priority: <span className="badge badge-
success">{this.state.priority}</span></small>
                    </a>
                </div>
            <br/><br/>
            <h4>Comments
                <div className="btn-group btn-group-sm float-right" role="group" aria-
label="Basic example">
                    <button type="button" className="btn btn-outlineprimary btn-sm">Add
Comment</button>
                </div>
            </h4>
        </div>
    </div>
);
}
}

```

INCIDENTS ROUTES

```
const router = require('express').Router();
let Incident = require('../models/incident.model');

//Get all incidences
router.route('/all').get((req,res)=>{
  Incident.find()
    .then(incidents=>res.json(incidents))
    .catch(err=>res.status(400).json("Error:"+err));
});

//Get all incidences not closed router.route('/').get((req,res)=>{
  Incident.find().where("status").in(["new",'dispatched','progress'])
    .then(incidents=>res.json(incidents))
    .catch(err=>res.status(400).json("Error:"+err))
});

//get by id
router.route('/:id').get((req,res)=>{
  Incident.findById(req.params.id)
    .then(incident=>res.json(incident))
    .catch(err=>res.status(400).json("Error:"+err));
});

//Add incident
router.route('/add').post((req,res)=>{  const number =
req.body.number;                      const  description  =
req.body.description;                 const  customer_name =
req.body.customer_name;               const  priority    =
req.body.priority;                    const  narrative   =
req.body.narrative;
```

```
const newIncident = new
Incident({number,description,customer_name,priority,narrative});  newIncident.save()
  .then()=>res.json('Incident added!'))
  .catch(err=>res.status(400).json("Error:"+err));
});

//Update incident
router.route('/update/:id').post((req,res)=>{
  Incident.findById(req.params.id)
  .then(exercise=>{
    if(req.body.description != null)
      exercise.description = req.body.description;          if(req.body.priority != null)
      exercise.priority = req.body.priority;          if(req.body.narrative != null)
      exercise.narrative = exercise.narrative + ";" + req.body.narrative;

    exercise.save()
      .then()=>res.json("Incident updated!"))
      .catch(err=>res.status(400).json("Error updating record!" + err));    })
    .catch(err=>res.status(400).json("Error!" + err));
  });

//Delete incident
router.route('/:id').delete((req,res)=>{  Incident.findByIdAndDelete(req.params.id)
  .then()=>res.json("Incident deleted!"))
  .catch(err=>res.status(400).json("Error deleting record:" + err));
});

module.exports = router;
```

BACKEND SERVER

```
const express = require('express'); const cors =
require('cors');

const mongoose = require('mongoose');

require('dotenv').config();

const app = express();
const port = process.env.PORT || 5000; app.use(cors());
app.use(express.json());

const uri = process.env.ATLAS_URI;
mongoose.connect(uri,{ useNewUrlParser:true,useCreateIndex:true });

const connection = mongoose.connection; connection.once('open',()=>{
  console.log("MongoDB database connection established successfully");
});

//>>>Model routers
const  userRouter  =  require('./routes/users');  const
incidentRouter  =  require('./routes/incidents');  const
commentRouter = require('./routes/comments');

app.use('/users',userRouter);          app.use('/incidents',incidentRouter);
app.use('/comments',commentRouter);

//<<<<<<model routers

app.listen(port,()=>{
  console.log(`Server is running on port: ${port} `); });
```

FUTURE FUNCTIONALITIES

Customer login – Create accounts for customers

User editing, and delete – Add CRUD functionality for users

Incident deleting, updating, logs – Enable incident deleting, update and keeping of logs as a separate table showing who did what.

User roles matrix – Define roles and rights for users

PROJECT URL

Azure Link: - <https://incidentsite.azurewebsites.net>

GitHub Repo Link: - <https://github.com/anish25/ExtraSeriousCompany>