ABSTRACT

Teaching, now a days, has become a challenging and ever-changing profession. It is transforming in every aspects. Teaching must be done in accordance with the learner's needs. A teacher is expected to follow a plan of coverage of syllabus given by external entity and record the speed bumps that she/he encounters while doing so. So, a teacher has to plan accordingly to finish her teaching in the given period of time. If for some reason there exists any delay between the proposed and actual coverage time, this should be solved via extra classes for the said course but the faculty should take the time-table into account while specifying extra hours. It will help the teacher in avoiding bad practices implemented by a teacher unintentionally and at the same time makes the learning process effective and comfortable to the learner. It can be done by meticulous planning and careful generating a report for plan of coverage of syllabus in the beginning of every semester by teacher before taking a class. In all, a Teaching Report is a great help in this regard.

Features:

Teacher Enter Details:

Any faculty can register through register from to get their schedule of plan of coverage of syllabus which help them to plan all the learning process effective and comfortable to the learner.

Report Generation

Any faculty who register for the teaching diary can fill the details through the teaching form through which they can generate the teaching diary report which include the details of respective name of the faculty, course code, course name, credits and proposed plan of coverage of syllabus according to the term.

TABLE OF CONTENTS

INTRODUCTION

In much modern usage, the words ‘teaching’ and ‘teacher’ are wrapped up with schooling and schools. One way of approaching the question ‘What is teaching?’ is to look at what those called ‘teachers’ do – and then to draw out key qualities or activities that set them apart from others. The problem is that all sorts of things are bundled together in job descriptions or roles that may have little to do with what we can sensibly call teaching.

Teaching and its methodology have seen some ground breaking shifts, further more in past decade or so. As the need of giving right education in most efficient way possible has given rise or rather need to reduce a bit of manpower from the people who are directly or indirectly are involved in this discipline. And nowhere it’s more apparent than institutions for higher studies where practices to improve upon the way and what needs to be taught changes more often if not regularly.

Handling everyday routines of college or any institution is one of those tasks that may use a helping hand of technology. One shouldn’t say that all the functionalities or everyday task should be automated, but one can at least try to ease up the burden of teachers who already have a lot on their plate to deal with.

One such issue came to light when it became tedious to manage the class allocation and other basic tasks related to number of classes that needed to be taken and permutations and combinations that enveloped the system itself. In a credit-based education system it’s of utmost significance on how much classes needs to be taken and if due to any reason number of days are less than number of classes, then it becomes an issue.

Teaching Diary handles the aforesaid issue along with several others. It not only allows each and every faculty to manage and take in accordance to their and institution and more importantly student’s ease. But even goes further in managing small tasks such as taking internals and managing lab records.

1.1 OVERVIEW:

Before this application came into light, the functionalities of Teaching Report were done manually by the respective faculties although it was fairly easy task but still to maintain the classes and records for each semester especially when there is fair amount of changes in the subjects and its credits. It become that much difficult to co-relate each and every class per subject. If due to any reason there is difference in number of classes and number of days remaining due to external factors, earlier it was hard to keep track of all such occurrences.

Now with the proposed application, Teaching Report is to maintain all records of the classes held by a particular teacher where each teacher will be prompted to enter the details such as, Department Name, Semester, Section, Batch, Term, Course Code and Name, Name of the faculty, course Credits and total session required which would be calculated by credits associated to each subject. And along with this teacher should select the proposed plan of class hours allocated which shall be followed along with the dates and timing provided.

After entering all these above-mentioned details teacher will follow the scheduled coverage of topics according to the plan selected and the dates and timing which would be reflected in the generated report. If in case the teacher is not able to follow the plan with the syllabus which needs to be covered by that duration, then they have to take special classes in order to complete the portions which they have planned keeping in their respective durations of class hours allocated.

1.2 PROPOSED SYSTEM (Problem Definition):

The problem in the existing: -

The issue with current way of teaching in class which involves all manual work is a burden for teachers where they won't be able to get the count of classes along with the coverage of topic and timings which is selected from the time-table. There is a need for a systematic approach in the way of organizing the work and the completion within the allocated slot.

This is particularly useful in the case of learning or teaching plan which was proposed. By this there would be a way of knowing which of the teachers were not able to complete their portions and where they have to include extra classes. Moreover, they must be informed where they have to improve their teaching skills.

2. FUNCTIONAL REQUIRNMENTS

2.1 HARDWARE REQUIRNMENTS:

Any PC, mobile or tablet computer that can run a web browser and is capable of connecting to the internet.

2.2 SOFTWARE REQUIRNMENTS:

2.2.1 For users:

Browser:	Any modern web browser.
	Ideal when viewed on Firefox.

2.2.2 For developers:

Operating System:	Microsoft Windows 8.1
Front End:	React
Server-Side:	Node.js, Express.js
Back End:	MongoDB
Browser:	Google Chrome

3. SOFTWARE REQUIRNMENTS SPECIFICATION

This document that describes the nature of a project, software or application. In simple words, this document is manual of a project provided it is prepared before you start a project/application. This document is also known as SRS report, software report. A software document is primarily prepared or a project, software or any kind of application.

3.1 Overall Description:

3.1.1 Product Perspective

In its entirety application itself is uses basic functionalities of create, read and generate operations. Any user can generate as many reports. It is developed in order to help faculties in maintaining their schedules.

3.1.2 Product Functions:

3.1.2.1 Generate Report

In the following function user (Faculty) will enter the following credentials: -

1) Department Name
2) Semester
3) Section
4) Batch
5) Term
6) Course Code and Name
7) Name of the Faculty
8) Course Credits
9) Total Session Required (This field will be calculated on the basis of credits specified for the said course)

After entering the afore mentioned details by the user(faculty) for the following function, the time-table for the specified semester, course, term will be imported from the database which will be available in the PDF format.

3.2 User Classes and Characteristics

For the following software the faculties would be the people who register themselves for the application.

3.3 System Features

It would provide the descriptive documentation of the for each feature supported by the following software. It would further have all the module listed along with any submodules and fields that defines the feature itself.

3.3.1 Register: - This module would help the faculties to register to generate the teaching report.

- **Input:** Add personal details such as name, email id and password.
- **Output:** User id and password for login

3.3.2 Login: - To get into the Teaching module they need to enter the user's credentials and it should match with the database.

- **Input:** User need to enter their credentials to login.
- **Output:** Users will redirect to the in the main page if users are credentials are match with the database or it will display an error.

3.3.3 Teaching Detail Form: -

This module would help the user to add the details of their respective subjects along with the other details.

- **Input:** Provide all the details of the Teacher.
- **Output:** Details will be stored into database.

3.3.4 Generate Report: -This module would help the user to generate a pdf report from the details filled by the faculties

- **Input:** Select the button to generate a pdf report.

4. IMPLIMENTATION

4.1 Teaching Module

```
import React, {Component} from 'react'
import axios from 'axios'
import {Redirect} from 'react-router-dom'
import "../../styles/TeachingDiary.css"
import { AuthContext } from '../../Context/Context';

class TeachingDiary extends Component{

  state = {
    department: '',
    semester: '',
    section: '',
    fromterm: '',
    toterm: '',
    batch: '',
    subject: '',
    credits: '',
    facultyname: '',
    totalhours: '',
    mon: [],
    tue: [],
    wed: [],
    thu: [],
    fri: [],
    sat: [],
    isSubmitted: null
  }


  handleFormChange = (e) => {

    console.log('handleFormChange', e.target.type);

    if (e.target.type == "checkbox") {
      if (e.target.checked) {
        switch (e.target.name) {
          case "mon":
            this.setState({
              mon: [...this.state.mon, e.target.value]
            })
            break;

          case "tue":
            this.setState({
              tue: [...this.state.tue, e.target.value]
            })
```

```
            break;

        case "wed":
            this.setState({
                wed: [...this.state.wed, e.target.value]
            })
            break;

        case "thu":
            this.setState({
                thu: [...this.state.thu, e.target.value]
            })
            break;

        case "fri":
        this.setState({
            fri: [...this.state.fri, e.target.value]
            })
            break;

        case "sat":
            this.setState({
                sat: [...this.state.sat, e.target.value]
            })
            break;

        default:
            break;
    }
}
else {
    switch (e.target.name) {
        case "mon":
            let monupdate = this.state.mon.filter(val => val != e.target.value)
            this.setState({
                mon: monupdate
            })
            break;

        case "tue":
            let tueupdate = this.state.tue.filter(val => val != e.target.value)
            this.setState({
                tue: tueupdate
            })
            break;

        case "wed":
            let wedupdate = this.state.wed.filter(val => val != e.target.value)
            this.setState({
                wed: wedupdate
            })
```

```
                break;

            case "thu":
                let thuupdate = this.state.thu.filter(val => val != e.target.value)
                this.setState({
                    thu: thuupdate
                })
                break;

            case "fri":
                let friupdate = this.state.fri.filter(val => val != e.target.value)
                this.setState({
                    thu: friupdate
                })
                break;

            case "sat":
                let satupdate = this.state.sat.filter(val => val != e.target.value)
                this.setState({
                    thu: satupdate
                })
                break;

            default:
                break;
        }
      }
    } else {
      this.setState({
        [e.target.name]: e.target.value
      })
    }
}

submitUser = (e) => {
    e.preventDefault()

    axios.post('/teaching/addteachingdiary', {

      semester: this.state.semester,
      section: this.state.section,
      fromterm: this.state.fromterm,
      department: this.state.department,
      batch: this.state.batch,
      toterm: this.state.toterm,
      subject: this.state.subject,
      credits: this.state.credits,
      facultyname: this.state.facultyname,
      totalhours: this.state.totalhours,
      days: {
        mon: this.state.mon,
```

```
            tue: this.state.tue,
            wed: this.state.wed,
            thu: this.state.thu,
            fri: this.state.fri,
            sat: this.state.sat
          },
          username: sessionStorage.getItem('username')
        })

        .then(res => {
          if(res.data)
            this.setState({isSubmitted: true})
          else
            this.setState({isSubmitted: false})
        })
        .catch(err => console.log(err))

    }

    static contextType = AuthContext
    render() {

        if(!sessionStorage.getItem('username'))
          return <Redirect to= "/"/>

        if(this.state.isSubmitted)
          return <Redirect to="teachingtiming" />

        const {selected_dept} = this.context

        return(
          <div>
            <center>
              <h6 className="teaching">Department of Computer pplication</h6>
              <h6 className="teaching">GENERATE TEACHING DIARY</h6>

              <h3>{selected_dept}</h3>
            </center>
            <div className="first_div row">
              <form onSubmit={e => this.submitUser(e)}>
              <div className="row">
                <div className="input-field col s4 ">
                  <label className="diary_text">Name of the faculty: </label>
                  <input type="text" name="facultyname" onChange={this.handleFormChange}
/>
                </div>
                <div className="input-field col s4">
                  <label className="diary_text">Semester: </label>
                  <input type="text" name="semester" onChange={this.handleFormChange} />
                </div>
                <div className="input-field col s4">
```

```
                    <label className="diary_text">Department: </label>
                    <input type="text" name="department" onChange={this.handleFormChange}
/>
                  </div>
                </div>
                <div className="row">
                  <div className="input-field col s4">
                    <label className="diary_text">Section: </label>
                    <input type="text" name="section" onChange={this.handleFormChange} />
                  </div>
                  <div className="input-field col s4">
                    <label className="diary_text">Subject: </label>
                    <input type="text" name="subject" onChange={this.handleFormChange} />
                  </div>
                  <div className="input-field col s4">
                    <label className="diary_text">Batch: </label>
                    <input type="text" name="batch" onChange={this.handleFormChange} />
                  </div>
                </div>
                <div className="row">
                  <div className="input-field col s4">
                    <label className="diary_text">FromTerm: </label>
                    <input type="date" name="fromterm" onChange={this.handleFormChange} />
                  </div>
                  <div className="input-field col s4">
                    <label className="diary_text">To Term: </label>
                    <input type="date" name="toterm" onChange={this.handleFormChange} />
                  </div>
                  <div className="input-field col s5">
                    <label className="diary_text">Credits: </label>
                    <input type="text" name="credits" onChange={this.handleFormChange} />
                  </div>
                  <div className="input-field col s5">
                    <label className="diary_text">Total no. of hours: </label>
                    <input type="text" name="totalhours" onChange={this.handleFormChange} />
                  </div>
                </div>

                    {/* Time Table */}
                    <br/>
                    <div>
                      <table>
                      <tbody>
                        <tr>
                          <th></th>
                          <th>9:00-9:55</th>
                          <th>9:55-10:50</th>
                          <th>11:05-12:00</th>
                          <th>12:00-12:55</th>
                          <th>1:45-2:40</th>
                          <th>2:40-3:35</th>
```

```
                        <th>3:55-4:30</th>
                      </tr>
                      <tr>
                        <td>Monday</td>

                        <td>
                          <div className="table_checkbox">
                            <label className="valign-wrapper">
                              <input type="checkbox" className="filled-in" name="mon"
value="9:00-9:55" onChange={this.handleFormChange} />
                              <span></span>
                            </label>
                          </div>
                        </td>
                        <td>
                          <div className="table_checkbox">
                            <label className="valign-wrapper">
                              <input type="checkbox" className="filled-in" name="mon"
value="9:55-10:50" onChange={this.handleFormChange}/>
                              <span></span>
                            </label>
                          </div>
                        </td>
                        <td>
                          <div className="table_checkbox">
                            <label className="valign-wrapper">
                              <input type="checkbox" className="filled-in" name="mon"
value="11:05-12:00" onChange={this.handleFormChange}/>
                              <span></span>
                            </label>
                          </div>

                        </td>
                        <td>
                          <div className="table_checkbox">
                            <label className="valign-wrapper">
                              <input type="checkbox" className="filled-in" name="mon"
value="12:00-12:55" onChange={this.handleFormChange}/>
                              <span></span>
                            </label>
                          </div>
                        </td>
                        <td>
                          <div className="table_checkbox">
                            <label className="valign-wrapper">
                              <input type="checkbox" className="filled-in" name="mon"
value="1:45-2:40" onChange={this.handleFormChange}/>
                              <span></span>
                            </label>
                          </div>
                        </td>
```

```
                        <td>
                          <div className="table_checkbox">
                            <label className="valign-wrapper">
                              <input type="checkbox" className="filled-in" name="mon"
value="2:40-3:35" onChange={this.handleFormChange}/>
                              <span></span>
                            </label>
                          </div>
                        </td>
                        <td>
                          <div className="table_checkbox">
                            <label className="valign-wrapper">
                              <input type="checkbox" className="filled-in" name="mon"
value="3:55-4:30" onChange={this.handleFormChange}/>
                              <span></span>
                            </label>
                          </div>
                        </td>
                      </tr>
                      <tr>
                        <td>Tuesday</td>
                        <td>
                          <div className="table_checkbox">
                            <label className="valign-wrapper">
                              <input type="checkbox" className="filled-in" name="tue"
value="9:00 - 9:55" onChange={this.handleFormChange}/>
                              <span></span>
                            </label>
                          </div>
                        </td>
                        <td>
                          <div className="table_checkbox">
                            <label className="valign-wrapper">
                              <input type="checkbox" className="filled-in" name="tue"
value="9:55-10:50" onChange={this.handleFormChange}/>
                              <span></span>
                            </label>
                          </div>
                        </td>
                        <td>
                          <div className="table_checkbox">
                            <label className="valign-wrapper">
                              <input type="checkbox" className="filled-in" name="tue"
value="11:05-12:00" onChange={this.handleFormChange}/>
                              <span></span>
                            </label>
                          </div>
                        </td>
                        <td>
                          <div className="table_checkbox">
                            <label className="valign-wrapper">
```

```
                                    <input type="checkbox" className="filled-in" name="tue"
value="12:00-12:55" onChange={this.handleFormChange}/>
                                    <span></span>
                                </label>
                            </div>
                        </td>
                        <td>
                            <div className="table_checkbox">
                                <label className="valign-wrapper">
                                    <input type="checkbox" className="filled-in" name="tue"
value="1:45-2:40" onChange={this.handleFormChange}/>
                                    <span></span>
                                </label>
                            </div>
                        </td>
                        <td>
                            <div className="table_checkbox">
                                <label className="valign-wrapper">
                                    <input type="checkbox" className="filled-in" name="tue"
value="2:40-3:35" onChange={this.handleFormChange}/>
                                    <span></span>
                                </label>
                            </div>
                        </td>
                        <td>
                            <div className="table_checkbox">
                                <label className="valign-wrapper">
                                    <input type="checkbox" className="filled-in" name="tue"
value="3:55-4:30" onChange={this.handleFormChange}/>
                                    <span></span>
                                </label>
                            </div>
                        </td>
                    </tr>
                    <tr>
                        <td>Wensday</td>
                        <td>
                            <div className="table_checkbox">
                                <label className="valign-wrapper">
                                    <input type="checkbox" className="filled-in" name="wed"
value="9:00-9:55" onChange={this.handleFormChange}/>
                                    <span></span>
                                </label>
                            </div>
                        </td>
                        <td>
                            <div className="table_checkbox">
                                <label className="valign-wrapper">
                                    <input type="checkbox" className="filled-in" name="wed"
value="9:55-10:50" onChange={this.handleFormChange}/>
                                    <span></span>
```

```
                                    </label>
                                </div>
                            </td>
                            <td>
                                <div className="table_checkbox">
                                    <label className="valign-wrapper">
                                        <input type="checkbox" className="filled-in" name="wed"
value="11:05-12:00" onChange={this.handleFormChange}/>
                                        <span></span>
                                    </label>
                                </div>
                            </td>
                            <td>
                                <div className="table_checkbox">
                                    <label className="valign-wrapper">
                                        <input type="checkbox" className="filled-in" name="wed"
value="12:00-12:55" onChange={this.handleFormChange}/>
                                        <span></span>
                                    </label>
                                </div>
                            </td>
                            <td>
                                <div className="table_checkbox">
                                    <label className="valign-wrapper">
                                        <input type="checkbox" className="filled-in" name="wed"
value="1:45-2:40" onChange={this.handleFormChange}/>
                                        <span></span>
                                    </label>
                                </div>
                            </td>
                            <td>
                                <div className="table_checkbox">
                                    <label className="valign-wrapper">
                                        <input type="checkbox" className="filled-in" name="wed"
value="2:40-3:35" onChange={this.handleFormChange}/>
                                        <span></span>
                                    </label>
                                </div>
                            </td>
                            <td>
                                <div className="table_checkbox">
                                    <label className="valign-wrapper">
                                        <input type="checkbox" className="filled-in" name="wed"
value="3:55-4:30" onChange={this.handleFormChange}/>
                                        <span></span>
                                    </label>
                                </div>
                            </td>
                        </tr>
                        <tr>
                            <td>Thursday</td>
```

```
                        <td>
                          <div className="table_checkbox">
                            <label className="valign-wrapper">
                              <input type="checkbox" className="filled-in" name="thu"
value="9:00 - 9:55" onChange={this.handleFormChange}/>
                              <span></span>
                            </label>
                          </div>
                        </td>
                        <td>
                          <div className="table_checkbox">
                            <label className="valign-wrapper">
                              <input type="checkbox" className="filled-in" name="thu"
value="9:55-10:50" onChange={this.handleFormChange}/>
                              <span></span>
                            </label>
                          </div>
                        </td>
                        <td>
                          <div className="table_checkbox">
                            <label className="valign-wrapper">
                              <input type="checkbox" className="filled-in" name="thu"
value="11:05-12:00" onChange={this.handleFormChange}/>
                              <span></span>
                            </label >
                          </div>
                        </td>
                        <td>
                          <div className="table_checkbox">
                            <label className="valign-wrapper">
                              <input type="checkbox" className="filled-in" name="thu"
value="12:00-12:55" onChange={this.handleFormChange}/>
                              <span></span>
                            </label>
                          </div>
                        </td>
                        <td>
                          <div className="table_checkbox">
                            <label className="valign-wrapper">
                              <input type="checkbox" className="filled-in" name="thu"
value="1:45-2:40" onChange={this.handleFormChange}/>
                              <span></span>
                            </label>
                          </div>
                        </td>
                        <td>
                          <div className="table_checkbox">
                            <label className="valign-wrapper">
                              <input type="checkbox" className="filled-in" name="thu"
value="2:40-3:35" onChange={this.handleFormChange}/>
                              <span></span>
```

```
                                    </label>
                                </div>
                            </td>
                            <td>
                                <div className="table_checkbox">
                                    <label className="valign-wrapper">
                                        <input type="checkbox" className="filled-in" name="thu"
value="3:55-4:30" onChange={this.handleFormChange}/>
                                        <span></span>
                                    </label>
                                </div>
                            </td>
                        </tr>
                        <tr>
                            <td>Friday</td>
                            <td>
                                <div className="table_checkbox">
                                    <label className="valign-wrapper">
                                        <input type="checkbox" className="filled-in" name="fri"
value="9:00 - 9:55" onChange={this.handleFormChange}/>
                                        <span></span>
                                    </label>
                                </div>
                            </td>
                            <td>
                                <div className="table_checkbox">
                                    <label className="valign-wrapper">
                                        <input type="checkbox" className="filled-in" name="fri"
value="9:55-10:50" onChange={this.handleFormChange}/>
                                        <span></span>
                                    </label>
                                </div>
                            </td>
                            <td>
                                <div className="table_checkbox">
                                    <label className="valign-wrapper">
                                        <input type="checkbox" className="filled-in" name="fri"
value="11:05-12:00" onChange={this.handleFormChange}/>
                                        <span></span>
                                    </label>
                                </div>
                            </td>
                            <td>
                                <div className="table_checkbox">
                                    <label className="valign-wrapper">
            </div>
        )
    }
}

export default TeachingDiary
```

4.2.2 Teaching Diary Model

```
const router = require('express').Router()
const TeachingModel = require('../models/teaching.model')

// For Adding Faculities Details

router.post('/addteachingdiary', (req,res) => {
    console.log("Inserted")

    const newTeaching = new TeachingModel({
        semester: req.body.semester,
        section: req.body.section,
        department: req.body.department,
        batch: req.body.batch,
        fromterm: req.body.fromterm,
        toterm: req.body.toterm,
        subject: req.body.subject,
        credits: req.body.credits,
        facultyname: req.body.facultyname,
        totalhours : req.body.totalhours,
        days: req.body.days,
        username: req.body.username
    })

    newTeaching.save()
    .then(() => res.send(true))
    .catch(err => res.send(false))

})

router.get('/getTeachingDetails/:username', (req, res) => {
    console.log('loggedin username: ', req.params.username)
    TeachingModel.find({username: req.params.username}).sort({_id: -1}).limit(1)
        .then(data => {
            console.log('data from getTeachingDetails: ', data)
            res.send(data)
        })
        .catch(err => res.send(err))
})

module.exports = router
```

4.3 Teaching Timing Module

```
import React, {Component, Fragment} from 'react'
import axios from 'axios'
import {Redirect} from 'react-router-dom'
import "../../styles/TeachingTiming.css"
import html2canvas from 'html2canvas'
import * as jsPDF from 'jspdf'

class TeachingTiming extends Component {

    state = {
        department: '',
        semester: '',
        section: '',
        batch: '',
        fromterm: '',
        toterm: '',
        credits: '',
        subject: '',
        facultyname: '',
        totalhours: '',
        date_and_timing: [],
        holidays: [
            'Wed May 01 2019',
            'Tue Jan 01 2019',
            'Sat Jan 26 2019',
            'Mon Mar 04 2019',
            'Wed Mar 20 2019'
        ]
    }

    componentDidMount() {
        //localhost:3000/teaching/Teachingdiary/negi
        axios.get('/teaching/getTeachingDetails/' + sessionStorage.getItem('username'))
            .then(data => {

            console.log('data: teaching timing: ', data.data)
                this.handleSetState(data.data)

            })
            .catch(err => console.log(err))
    }

    handleSetState = (data) => {
        let fromterm = new Date(data[0].fromterm),
            toterm = new Date(data[0].toterm),
            daylist = [],
            store_date = [],
            date_and_timing = []
```

```
for(var dt=fromterm; dt<=toterm; dt.setDate(dt.getDate()+1)){
   daylist.push(new Date(dt));
}

daylist.forEach(v => {
   if(v.getDay() !== 0)
   {
      store_date = [...store_date, v.toDateString()]
   }
      // store_date = [...store_date, v.toISOString().slice(0,10)]

})

store_date.forEach(dt => {

   switch (dt.slice(0,3)) {
      case "Mon":
         date_and_timing = [...date_and_timing, {date: dt, time: data[0].days.mon}]
         break;

      case "Tue":
         date_and_timing = [...date_and_timing, {date: dt, time: data[0].days.tue}]
         break;

      case "Wed":
         date_and_timing = [...date_and_timing, {date: dt, time: data[0].days.wed}]
         break;

      case "Thu":
         date_and_timing = [...date_and_timing, {date: dt, time: data[0].days.thu}]
         break;

      case "Fri":
         date_and_timing = [...date_and_timing, {date: dt, time: data[0].days.fri}]
         break;

      case "Sat":
         date_and_timing = [...date_and_timing, {date: dt, time: data[0].days.sat}]
         break;

      default:
         break;
   }
})

console.log('teaching time :',data)

this.setState({
   semester: data[0].semester,
   date_and_timing,
   fromterm: data[0].fromterm,
```

```
            toterm: data[0].toterm,
            credits: data[0].credits,
            department: data[0].department,
            section: data[0].section,
            batch: data[0].batch,
            term: data[0].term,
            subject: data[0].subject,
            facultyname: data[0].facultyname,
            totalhours: data[0].totalhours,
        })

    }

    convertDomToPdf = () => {
        console.log("came inside button")
        const input = document.getElementById('div_to_print');
    html2canvas(input)
      .then((canvas) => {
        const imgData = canvas.toDataURL('image/png');
        const pdf = new jsPDF();
        pdf.addImage(imgData, 'JPEG', 0, 0);
        // pdf.output('dataurlnewwindow');
        pdf.save("download.pdf");
      })
    }

    render() {

        if(!sessionStorage.getItem('username'))
            return <Redirect to="/"/>

        return(

            <Fragment>

                <div id="div_to_print" style={{
                    backgroundColor: 'white',
                    border: '1px solid black',
                    width: '58%',
                    minHeight: '297mm',
                    marginLeft: 'auto',
                    marginRight: 'auto'
                }}>

                <div id="pdfdiv">
                    <div className="Main-head">
                        <div className="left-div">
                            <img src={process.env.PUBLIC_URL + "/images/logo.png"}
    className="responsive-img logo-img" ></img>
                        </div>
                        <div className="mid-div center">
```

```
                Teaching Diary (Practical/Drawing)
              </div>
              <div className="right-div"></div>
            </div>
            <div className="bottom-div">
              <p>Department: {this.state.department}
              Semester: {this.state.semester}
              Term: {this.state.fromterm} to {this.state.toterm}</p>
              <p>Batch: {this.state.batch}
              Course code and Name: {this.state.subject} </p><p>Credits: {this.state.credits}</p>
              <p>Name(s) of the faculty: {this.state.facultyname}
Total No. of Session required: {this.state.date_and_timing.length}</p>

            </div>


            <div style={{padding: '0 1em'}}>

              <table className="timing-table" border='1'>
                <thead>
                  <tr>
                    <th colSpan="3">   Proposed plan of coverage of
Syllabus</th>
                    <th rowSpan="2">If not taken, Reasons</th>
                    <th colSpan="2">Actual Coverage of Syllabus</th>
                    <th>Remarks</th>
                    <th>Initials of Faculty</th>
                  </tr>
                  <tr>
                    <th>Lesson No.</th>
                    <th style={{textAlign: 'center' ,border: '1px solid #00000030' }}>Date</th>
                    <th style={{textAlign: 'center'}}>Time</th>
                    {/* <th rowSpan="2">If not taken, Reasons</th> */}
                    <th>Date</th>
                    <th>Time</th>
                                        </tr>
                </thead>
                <tbody>

                  {
                    this.state.date_and_timing.map((store, index) => {
                      return !this.state.holidays.includes(store.date) ?
                      <tr key={index}>
                        <td>
                          {index+1}
                        </td>
                        <td className="timing-td-table">
                          {store.date.slice(4,15)}
                        </td>
                        <td className="timing-td-table">
                          {store.time.join(" | ")}
```

```
                        </td>
                                                null
                    })
                }

                <tr>
                  <td colSpan="4">
                    <p>Signature of Faculty</p>
                    <p>Date: </p>
                  </td>
                  <td colSpan="4">
                    <p>Signature of HOD</p>
                    <p>Date: </p>
                  </td>
                </tr>
              </tbody>
            </table>

          </div>


          <p style={{textAlign: 'center',fontWeight: '600'}}>Final remarks</p>
          <div style={{display: 'flex', padding: '1em'}}>
            <div style={{flex: 1}}>
              <p>Signature of facutly</p>
              <p>Date: </p>
            </div>
            <div style={{flex: 1, borderLeft: '1px solid grey', padding: '0 1em'}}>
              <p>Signature of HOD</p>
              <p>Date: </p>
            </div>
          </div>
        </div>
        </div>

        <button onClick={this.convertDomToPdf}>convert to pdf</button>
      </Fragment>

    )

  }
}

export default TeachingTiming
```

5. SCREENSHOTS (REPORTS)

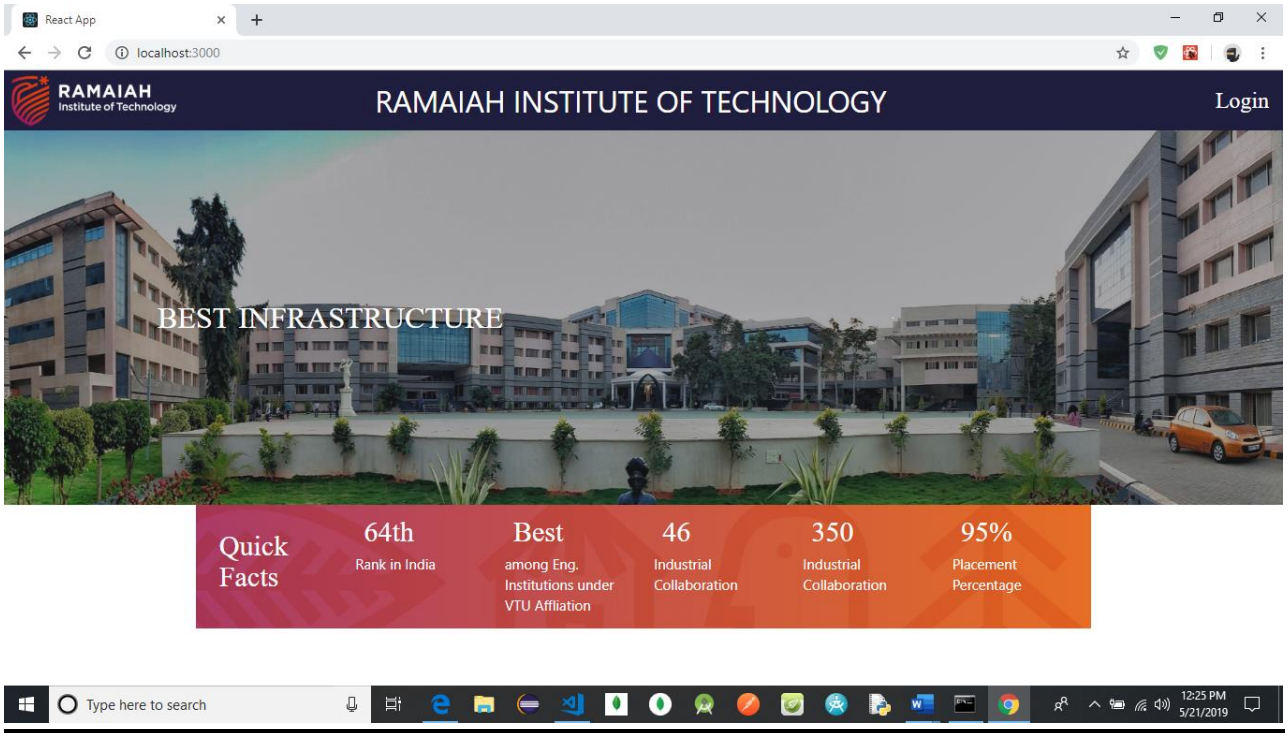

Figure 5.1

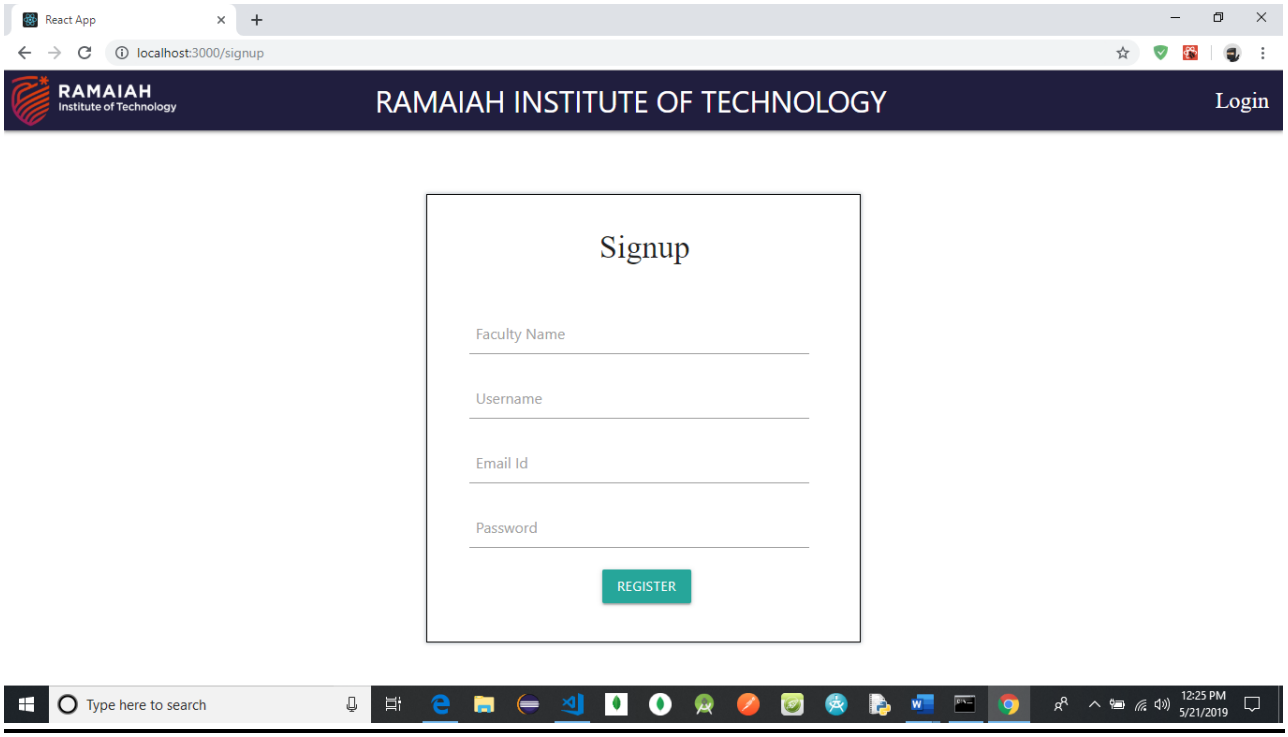

Figure 5.2

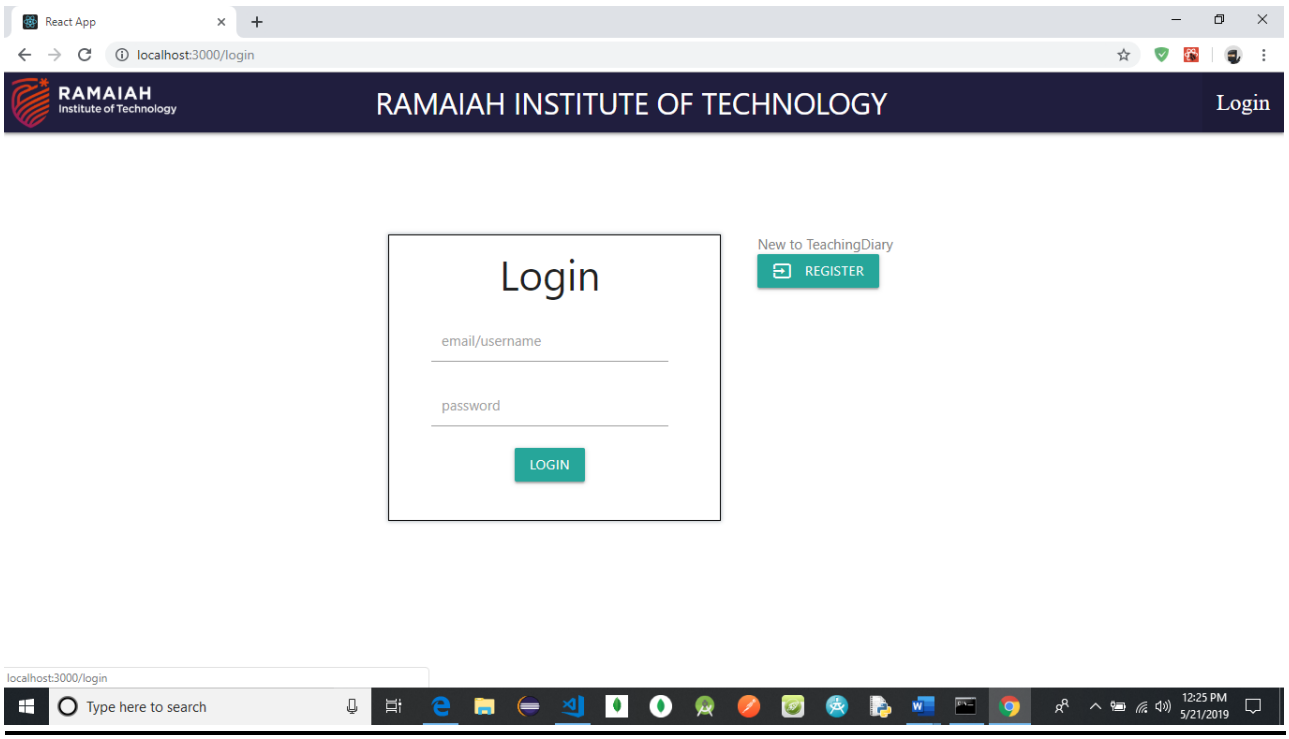

Figure 5.3

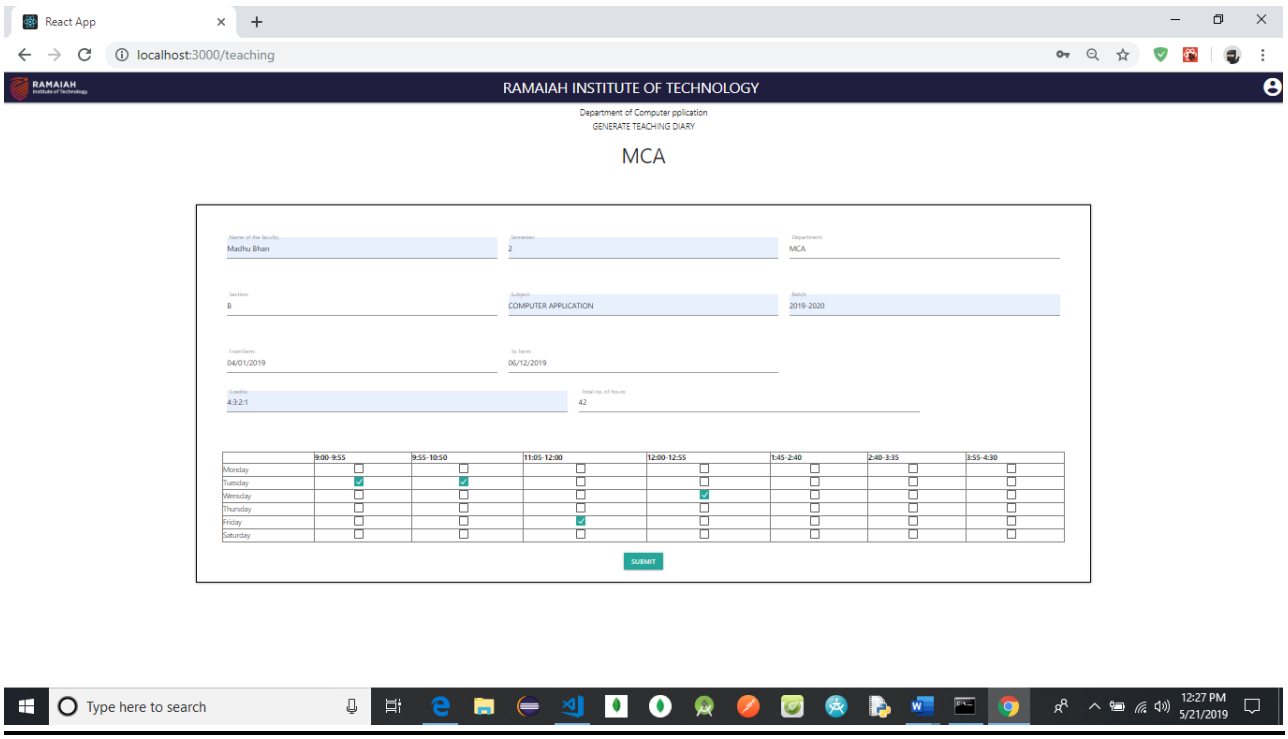

Figure 5.4

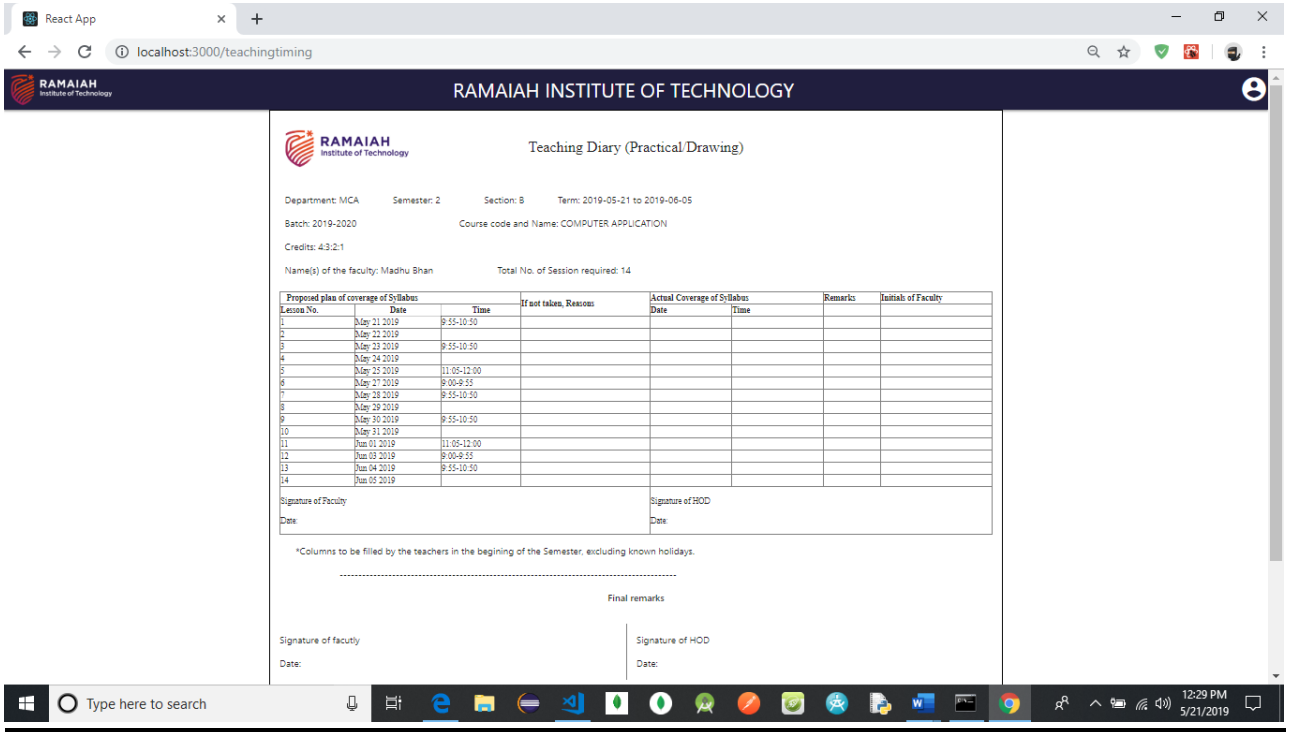
React App
localhost:3000/teachingtiming
RAMAIAH INSTITUTE OF TECHNOLOGY
RAMAIAH
Institute of Technology
Teaching Diary (Practical/Drawing)
Department: MCA Semester: 2 Section: B Term: 2019-05-21 to 2019-06-05
Batch: 2019-2020 Course code and Name: COMPUTER APPLICATION
Credits: 4:3:2:1
Name(s) of the faculty: Madhu Bhan Total No. of Session required: 14
Proposed plan of coverage of Syllabus
Lesson No.
Date
Time
If not taken, Reasons
Actual Coverage of Syllabus
Date
Time
Remarks
Initials of Faculty
1 May 21 2019 9:55-10:50
2 May 22 2019
3 May 23 2019 9:55-10:50
4 May 24 2019
5 May 25 2019 11:05-12:00
6 May 27 2019 9:00-9:55
7 May 28 2019 9:55-10:50
8 May 29 2019
9 May 30 2019 9:55-10:50
10 May 31 2019
11 Jun 01 2019 11:05-12:00
12 Jun 03 2019 9:00-9:55
13 Jun 04 2019 9:55-10:50
14 Jun 05 2019
Signature of Faculty
Date:
Signature of HOD
Date:
*Columns to be filled by the teachers in the begining of the Semester, excluding known holidays.
Final remarks
Signature of facutly
Date:
Signature of HOD
Date:
Type here to search
12:29 PM
5/21/2019

Figure 5.5

6. CONCLUSION

The pace at which science proceeds sometimes seems alarmingly slow, and impatience and hopes both run high when discussions turn to issues of learning and education. In the field of learning, the past quarter century has been a period of major research advances. Because of the many new developments, the studies that resulted in this volume were conducted to appraise the scientific knowledge base on human learning and its application to education. We evaluated the best and most current scientific data on learning, teaching, and learning environments. The objective of the analysis was to ascertain what is required for learners to reach deep understanding, to determine what leads to effective teaching, and to evaluate the conditions that lead to supportive environments for teaching and learning.

Meeting the learning needs of all students is a complex and demanding task for schools. How well students achieve at a school depends on factors such as how well teachers engage with their students, and the relationships schools have with their students' families. The assessment of student achievement, or understanding what students know and can do, is fundamental to effective teaching and to students' learning. Unless teachers know students well and are knowledgeable about their achievements, they cannot be confident that they are meeting the learning needs of their students.

In summary, students, teachers and school managers can use assessment information to improve learning only when they have:

- Collected good quality information that fairly represents what students know and can do.
- Analyzed the information to accurately determine the achievements of students;
- Correctly interpreted the information to report the achievements and progress of individuals and groups of students and to identify their next learning steps;
- Used the information to report to inform governance and management decision making.

7. FUTURE SCOPE AND ENHANCMENTS

Although the application helps in automating certain aspects of maintaining and generating reports about the classes and any discrepancies that involves dis similarities between the number of classes and days. But nothing is perfect in this world and so does this application therefore there can be some enhancements that would be even more user friendly and further automate teaching process entirely.

One such functionality that can further enhance the overall outlook of the application is its integration with gmail, how it would actually be implemented can be answered via some existing systems that already has this enabled as their feature for example fees receipt for MSRIT.

The scaling however is much easy to implement, as currently it is being used for single department. It is not hard to imagine it at a scale of college or university.

Upon registering each faculty will be given a unique hash value that can be used as a part of registering process in order to further enhance the email verification. Without verifying the email the faculty would be deprived of certain features until and unless he/she verifies the email.

8. BIBLIOGRAPHY

- **ReactJS** - https://reactjs.org/
- **Material-UI** - https://material-ui.com/
- **ExpressJS** - https://expressjs.com/