

Final Examination  
Year 3, Semester I

# **SE3040 – Application Frameworks**

Take-Home Exam

GitHub Repository

<https://github.com/nweerakeshara/ISPCompany.git>

Weerakeshara T.N.N.D.

IT18111552

q1.

### **Internet Packages Detail System**

There are several Electronic and Telecommunication companies in Sri Lanka. Sri Lanka Telecom, Dialog Axiata, Airtel Sri Lanka and Hutchison Telecommunication Lanka are some common examples.

Providing internet facilities and related infrastructure to their customers is one of the main services provided by those Electronic and Telecommunication companies. It is not just a service but also a main income earner for those companies. So, as internet service providers they try their best to provide their customers with a good service.

Normally telephone operators of those Electronic and Telecommunication companies get lot of calls from their customers asking details about the internet packages provided by them. Name of internet package, type of package, monthly rental, down payment, upload download speed and upload download data limits are some common questions they get every day. As a result, Management of those companies have to spend a big money and time on lot of telephone operators.

So, Electronic and Telecommunication companies can use this newly implemented Internet Package Detail System to solve this issue.

Let us take Airtel Sri Lanka as an example. Airtel management can assign one or two individuals from their company to add and manage internet packages details.

First of all, these employees have to register to the system. Then they can add and manage all the details of the internet packages. Customers that need to know about these internet packages can visit this online internet package detail system of Airtel to get details about internet packages. Here the customer can search packages by name or by their package type (4G, Broadband, Fiber).

The main objective from this Internet Packages Detail System is to minimize the number of calls a customer care of Electronic and Telecommunication company get asking for details about internet packages.

Most of the Electronic and Telecommunication companies have static web sites to display such details, but having an easily manageable dynamic site with a search to display internet packages details can be a great advantage to those companies.

## q2. a - Employee Register (Front End)

Register.component.js

```
import React, { Component } from "react";
import { connect } from "react-redux";
import PropTypes from "prop-types";
import { register } from "../actions/empActions";
import { clearErrors } from "../actions/errorActions";
import { Alert } from "reactstrap";
import swal from "sweetalert";

class RegisterEmp extends Component {
  state = {
    empUn: "",
    empEmail: "",
    empPw: "",
    empConfirmPw: "",
    empPin: "",
    msg: null,
  };

  static propTypes = {
    isAuthenticated: PropTypes.bool,
    error: PropTypes.object.isRequired,
    register: PropTypes.func.isRequired,
    clearErrors: PropTypes.func.isRequired,
  };

  registerClose = () => {
    this.props.clearErrors();
    this.setState({
      empUn: "",
      empEmail: "",
      empPw: "",
      empConfirmPw: "",
      empPin: "",
      msg: null,
      msgtop: null,
    });

    this.props.history.push("/addPackage");
  };

  componentDidUpdate = (prevProps) => {
    const { error, isAuthenticated } = this.props;
    if (error !== prevProps.error) {
      if (error.id === "REGISTER_FAIL") {
        this.setState({ msg: error.msg.msg });
      } else {
        this.setState({ msg: null });
      }
    }

    if (this.state.msg) {
      swal("Unsuccessful", this.state.msg, "error");
      this.setState({ msg: null });
    }
  }
}
```

```

    if (isAuthenticated) {
      this.registerClose();
    }
  };

  onChangeEmpUn = (e) => {
    this.setState({
      empUn: e.target.value,
    });
  };

  onChangeEmpEmail = (e) => {
    this.setState({
      empEmail: e.target.value,
    });
  };

  onChangeEmpPw = (e) => {
    this.setState({
      empPw: e.target.value,
    });
  };

  onChangeEmpPin = (e) => {
    this.setState({
      empPin: e.target.value,
    });
  };

  onChangeEmpConfirmPw = (e) => {
    if (e.target.value !== this.state.empPw) {
      this.setState({
        empConfirmPw: e.target.value,
        msgtop: "Confirm Password Does Not Match",
      });
    }
    if (e.target.value === this.state.empPw) {
      this.setState({
        empConfirmPw: e.target.value,
        msgtop: "",
      });
    }
  };

  onSubmit = (e) => {
    e.preventDefault();

    const { empUn, empEmail, empPw, empPin } = this.state;
    const newUser = {
      empUn,
      empEmail,
      empPw,
    };

    if (empPin === "55555") {
      this.props.register(newUser);

      this.setState({
        empUn: "",

```

```

        empEmail: "",
        empPw: "",
        empConfirmPw: "",
        empPin: "",
        msg: null,
        msgtop: null,
    });
} else {
    this.setState({
        empUn: "",
        empEmail: "",
        empPw: "",
        empConfirmPw: "",
        empPin: "",
        msg: null,
        msgtop: "Register Fail, Not an Employee",
    });
}
};

render() {
    return (
        <div style={{ marginTop: 10 }}>
            <h3>Employee Sign Up</h3>
            <form onSubmit={this.onSubmit}>
                {this.state.msgtop ? (
                    <Alert color="danger">{this.state.msgtop}</Alert>
                ) : null}
                <div className="form-group">
                    <label>Username :</label>
                    <input
                        type="text"
                        className="form-control"
                        value={this.state.empUn}
                        onChange={this.onChangeEmpUn}
                        maxLength="10"
                    />
                </div>

                <div className="form-group">
                    <label>Email Address :</label>
                    <input
                        type="email"
                        className="form-control"
                        value={this.state.empEmail}
                        onChange={this.onChangeEmpEmail}
                    />
                </div>

                <div className="form-group">
                    <label>Password :</label>
                    <input
                        type="password"
                        className="form-control"
                        value={this.state.empPw}
                        onChange={this.onChangeEmpPw}
                        minLength="5"
                    />
                </div>
            </form>
        </div>
    );
}

```

```

    <div className="form-group">
      <label>Confirm Password :</label>
      <input
        type="password"
        className="form-control"
        value={this.state.empConfirmPw}
        onChange={this.onChangeEmpConfirmPw}
        minLength="5"
      />
    </div>

    <div className="form-group">
      <label>PIN (To Confirm as a Real Employee) :</label>
      <input
        type="password"
        className="form-control"
        value={this.state.empPin}
        onChange={this.onChangeEmpPin}
      />
    </div>

    <div className="form-group">
      <input type="submit" value="Register" className="btn btn-primary" />
    </div>
  </form>
</div>
);
}
}

const mapStateToProps = (state) => ({
  isAuthenticated: state.emp.isAuthenticated,
  error: state.error,
});

export default connect(mapStateToProps, { register, clearErrors })(
  RegisterEmp
);

```

empActions.js (Used for Redux)

```

import
{USER_LOADING,USER_LOADED,REGISTER_FAIL,REGISTER_SUCCESS,LOGOUT_SUCCESS,LOGIN_FAIL
,LOGIN_SUCCESS,AUTH_ERROR} from "../types";
import {returnErrors} from "../errorActions";
import axios from 'axios';

```

```

////////////////////////////////////
//login
export const login = ({empUn , empPw}) => dispatch => {
  const config = {
    headers: {
      'Content-Type' : 'application/json'
    }
  }
  const body = JSON.stringify({empUn, empPw});

  axios.post('http://localhost:5000/api/emp/login', body, config).then(res =>
dispatch({
  type:LOGIN_SUCCESS,
  payload: res.data
})).catch(error => {
  dispatch(returnErrors(error.response.data, error.response.status,
'LOGIN_FAIL'));
  dispatch({
    type: LOGIN_FAIL
  }) ;
});
}

////////////////////////////////////
export const loadUser = () => (dispatch, getState) => {
  dispatch ({type: USER_LOADING});

  axios.get('http://localhost:5000/api/api/emp/get/emp',
tokenConfig(getState)).then(res => dispatch({ /*for token config*/
  type: USER_LOADED,
  payload: res.data
})).catch(error => {
  // dispatch(returnErrors(error.response.data, error.response.status));
  dispatch({
    type:AUTH_ERROR
  })
});
}

////////////////////////////////////
////////////////////////////////////
export const register = ({empUn, empEmail, empPw}) => dispatch => {
  const config = {
    headers: {
      'Content-Type' : 'application/json'
    }
  }
  const body = JSON.stringify({empUn,empEmail,empPw});

  axios.post('http://localhost:5000/api/emp/register', body, config).then(res =>

```

```

dispatch({ /*to register emp*/
  type:REGISTER_SUCCESS,
  payload: res.data
}).catch(error => {
  dispatch(returnErrors(error.response.data, error.response.status,
'REGISTER_FAIL'));
  dispatch({
    type: REGISTER_FAIL
  }) ;
});

}

/////////////////////////////////////////////////////////////////

export const logout = () => { /*logout*/
  return{
    type : LOGOUT_SUCCESS
  };
};

/////////////////////////////////////////////////////////////////

export const tokenConfig = (getState) => { /*token creation*/
  const token = getState().emp.token;
  const config = {
    headers : {
      "Content-type": "application/json"
    }
  }

  if(token){
    config.headers['emp_auth'] = token;
  }
  return config;
}

```

empReducer.js (Used for Redux)

```

import
{USER_LOADED,USER_LOADING,REGISTER_FAIL,REGISTER_SUCCESS,LOGIN_FAIL,LOGOUT_SUCCESS
,LOGIN_SUCCESS,AUTH_ERROR} from "../actions/types";

const initialState = {
  token : localStorage.getItem('token'),
  isAuthenticated : null,
  isLoading : false,
  user: null
};

export default function (state = initialState, action) {
  switch (action.type) {
    case USER_LOADING:
      return {

```



```

        ...state,
        isLoading: true
    };
    case USER_LOADED:
    return {
        ...state,
        isAuthenticated: true,
        isLoading: false,
        user: action.payload
    };

    case LOGIN_SUCCESS:
    case REGISTER_SUCCESS:
        localStorage.setItem('token', action.payload.token);
    return {
        ...state,
        ...action.payload,
        isAuthenticated: true,
        isLoading: false
    };

    case AUTH_ERROR:
    case LOGIN_FAIL:
    case LOGOUT_SUCCESS:
    case REGISTER_FAIL:
        localStorage.removeItem('token');
    return {
        ...state,
        token: null,
        user: null,
        isAuthenticated: false,
        isLoading: false
    };

    default : return state;
}
}

```

errorReducer.js (Used for Redux)

```

import
{USER_LOADED,USER_LOADING,REGISTER_FAIL,REGISTER_SUCCESS,LOGIN_FAIL,LOGOUT_SUCCESS
,LOGIN_SUCCESS,AUTH_ERROR} from "../actions/types";

const initialState = {
    token : localStorage.getItem('token'),
    isAuthenticated : null,
    isLoading : false,
    user: null
};

export default function (state = initialState, action) {

```

```

switch (action.type) {
  case USER_LOADING:
    return {
      ...state,
      isLoading: true
    };
  case USER_LOADED:
    return {
      ...state,
      isAuthenticated: true,
      isLoading: false,
      user: action.payload
    };

  case LOGIN_SUCCESS:
  case REGISTER_SUCCESS:
    localStorage.setItem('token', action.payload.token);
    return {
      ...state,
      ...action.payload,
      isAuthenticated: true,
      isLoading: false
    };

  case AUTH_ERROR:
  case LOGIN_FAIL:
  case LOGOUT_SUCCESS:
  case REGISTER_FAIL:
    localStorage.removeItem('token');
    return {
      ...state,
      token: null,
      user: null,
      isAuthenticated: false,
      isLoading: false
    };

  default : return state;
}
}

```

reducers/index.js (Used for Redux)

```

import {combineReducers} from "redux";
import empReducer from './empReducer';
import errorReducer from './errorReducer';

export default combineReducers({

  emp : empReducer,

```

```
    error : errorReducer
  });
```

## Employee Register (Back End)

routes/emp.js

```
////////////////////////////////////  
////////////////////////////////////  
//register emps  
router.post('/register', (req,res) =>{  
  
    //To register emp  
  
    const {empUn , empEmail, empPw} = req.body;  
    if(!empUn || !empEmail || !empPw){  
        return res.status(400).json({msg:'Please Fill All Fields'});  
    }  
  
    User.findOne({empUn}).then(user => {  
        if(user){  
            return res.status(400).json({msg : 'Username Already Exist'})  
        }  
    });  
  
    User.findOne({empEmail}).then(user => {  
        if(user){  
            return res.status(400).json({msg : 'Email Already Exist'})  
        }  
    });  
  
    const user = new User(req.body);  
    /*pw hashing*/  
    bcrypt.genSalt(10, (err, salt) => {  
  
        bcrypt.hash(user.empPw, saltRounds, function(error, hash) {  
            if (error) {  
                throw err;  
            }  
            user.empPw = hash;  
  
            user.save().then(user => {  
  
                jwt.sign(  
                    {_id : user._id}, "secret", {expiresIn: 10},  
                    (error, token) =>{  
                        if(error) {  
                            throw error;  
                        }  
                        res.json({  
                            token,  
                            user: {  
                                _id: user._id,  
                                empUn: user.empUn,
```

```
empEmail: user.empEmail
    }
  });
}
});
});
});
});
```

model/emp.js

```
const mongoose = require('mongoose');
const moment = require('moment');

const userSchema = mongoose.Schema({
  empUn: {
    type: String,
    maxlength: 50,
    unique: 1
  },
  empEmail: {
    type: String,
    trim: true,
    unique: 1
  },
  empPw: {
    type: String,
    minlength: 5
  },
  date : {
    type: Date,
    default: Date.now()
  },
  token : {
    type: String
  }
});
```

```
const User = mongoose.model('Employees', userSchema);
module.exports = {User};
```

## b - Employee Login and Logout (front end)

logout.component.js

```
import React, { Component } from "react";
import axios from "axios";
import { logout } from "../actions/empActions";
import { connect } from "react-redux";
import PropTypes from "prop-types";

class LogoutEmp extends Component {
  static propTypes = {
    logout: PropTypes.func.isRequired,
  };

  onClick = (e) => {
    e.preventDefault();
    this.props.logout();
  };

  render() {
    return (
      <div>

        <button
          className="btn btn-primary"
          onClick={this.onClick}
        >
          Logout
        </button>
      </div>
    );
  }
}

export default connect(null, { logout })(LogoutEmp);
```

login.component.js

```
import React, {Component} from "react";
import axios from "axios";
import {connect} from 'react-redux';
import PropTypes from 'prop-types';
import {login} from "../actions/empActions";
import {clearErrors} from "../actions/errorActions";
```

```

import swal from "sweetalert";
import {Link} from "react-router-dom";

class LoginEmp extends Component{

  state={
    empUn: "",
    empPw: "",
    msg :null
  }

  static propTypes = {
    isAuthenticated: PropTypes.bool,
    error : PropTypes.object.isRequired,
    login: PropTypes.func.isRequired,
    clearErrors : PropTypes.func.isRequired
  }

  loginClose = () => {
    this.props.clearErrors();
    this.setState({
      empUn: "",
      empPw: "",
      msg :null
    });

    this.props.history.push('/addPackage');
  }

  componentDidUpdate =(prevProps) => {
    const {error, isAuthenticated} = this.props;
    if(error !== prevProps.error){
      if(error.id === 'LOGIN_FAIL'){
        this.setState({msg : error.msg.msg});

      }else{
        this.setState({msg: null });
      }
    }

    if(this.state.msg){
      swal("Unsuccessful", this.state.msg, "error");
      this.setState({msg: null });
    }

    if(isAuthenticated){
      this.loginClose();
    }
  }

  onChangeEmpUn = (e) => {
    this.setState({
      empUn: e.target.value
    });
  }
}

```

```

onChangeEmpPw = (e) => {
  this.setState({
    empPw: e.target.value
  });
}

onSubmit = (e) => {
  e.preventDefault();

  const {empUn, empPw} = this.state;
  const existUser = {
    empUn,
    empPw
  }

  this.props.login(existUser);

  this.setState({
    empUn: "",
    empPw: "",
    msg :null
  });
}

render() {
  return (
    <div style={{marginTop: 10}}>
      <Link to={"/registerEmp"} className="nav-link"><button
className="btn btn-danger float-right">Employee Register</button></Link>
      <br/><br/>
      <h3>Employee Sign In</h3>
      <form onSubmit={this.onSubmit}>

        <div className="form-group">
          <label>Username :</label>
          <input type="text" className="form-control"
value={this.state.empUn} onChange={this.onChangeEmpUn}/>
        </div>

        <div className="form-group">
          <label>Password :</label>
          <input type="password" className="form-control"
value={this.state.empPw} onChange={this.onChangeEmpPw}/>
        </div>

        <div className="form-group">
          <input type="submit" value="Login" className="btn btn-
primary"/>
        </div>
      </form>
      <br/><br/><br/>
    </div>
  );
}

```

```

    }
  }

const mapStateToProps = state => ({
  isAuthenticated: state.emp.isAuthenticated,
  error : state.error
});

export default connect(mapStateToProps,{login, clearErrors})(LoginEmp);

```

**Here also empActions, errorActions, empReducer, errorReducer were used for redux. Their codes are pasted above with Employee Register.**

## Employee Login and Logout (back end)

routes/emp.js

```

////////////////////////////////////
////////////////////////////////////
//login
router.post('/login', (req,res) =>{

  //To login emp

  const {empUn , empPw} = req.body;
  if(!empUn || !empPw){
    return res.status(400).json({msg:'Please Fill All Fields'});
  }

  User.findOne({empUn}).then(user => {
    if(!user){
      return res.status(400).json({msg : 'Invalid Username'})
    }
    /*pw checking, hash passwords are checked here*/
    bcrypt.compare(empPw, user.empPw).then(result => {
      if(!result){
        return res.status(400).json({
          msg:'Invalid Credentials'
        });
      }

    }

    jwt.sign(
      {_id : user._id}, "secret", {expiresIn: 3500},
      (error, token) =>{
        if(error) {
          throw error;
        }
        res.json({
          token,
          user: {
            _id: user._id,
            empUn: user.empUn,

```



```

        empEmail: user.empEmail
      });
    }
  });
});

});

});

////////////////////////////////////
////////////////////////////////////

router.get('/get/emp', auth, (req, res) => {
  User.findById(req.user._id).select('-empPw').then(user => res.json(user));
});

module.exports = router;

```

model/emp.js

```

const mongoose = require('mongoose');

const moment = require('moment');

const userSchema = mongoose.Schema({

  empUn: {
    type: String,
    maxlength:50,
    unique: 1
  },

  empEmail: {
    type:String,
    trim: true,
    unique: 1
  },

  empPw: {
    type:String,
    minlength:5
  },

  date : {
    type:Date,
    default:Date.now()
  },

  token : {
    type: String
  }

```

```
})
```

```
const User = mongoose.model('Employees', userSchema);  
module.exports = {User};
```

## c - Add Package (front end)

addPackage.component.js

```
import React, {Component} from 'react';  
import axios from 'axios';  
import DefaultImg from './assets/default-img.jpg';  
import swal from "sweetalert";  
import { connect } from "react-redux";  
import PropTypes from "prop-types";  
import disableBrowserBackButton from "disable-browser-back-navigation";  
import Alert from "@material-ui/lab/Alert";  
  
class AddPackage extends Component{  
  state = {  
    serialCode: "",  
    packageName: "",  
    packageType: "",  
    monthlyCharge: "",  
    downloadSpeed: "",  
    uploadSpeed: "",  
    downloadLimit: "",  
    uploadLimit: "",  
    extraGBFee: "",  
    downPayment: "",  
    img1: "",  
    img2: "",  
    multerImage: DefaultImg,  
    msg: null  
  };  
  
  static propTypes = {  
    isAuthenticated: PropTypes.bool,  
    error: PropTypes.object.isRequired,  
    register: PropTypes.func.isRequired,  
    clearErrors: PropTypes.func.isRequired,  
  };  
  
  componentDidMount() {  
    disableBrowserBackButton();  
  }  
}
```

```
onChangeSerialCode =(e) =>{
  this.setState({
    serialCode: e.target.value
  });
}

onChangePackageName =(e) =>{
  this.setState({
    packageName: e.target.value
  });
}

onChangePackageType =(e) =>{
  this.setState({
    packageType: e.target.value
  });
}

onChangeMonthlyCharge =(e) =>{
  this.setState({
    monthlyCharge: e.target.value
  });
}

onChangeDownloadSpeed =(e) =>{
  this.setState({
    downloadSpeed: e.target.value
  });
}

onChangeUploadSpeed =(e) =>{
  this.setState({
    uploadSpeed: e.target.value
  });
}

onChangeDownloadLimit =(e) =>{
  this.setState({
    downloadLimit: e.target.value
  });
}

onChangeUploadLimit =(e) =>{
  this.setState({
    uploadLimit: e.target.value
  });
}

onChangeExtraGBFee =(e) =>{
  this.setState({
    extraGBFee: e.target.value
  });
}

onChangeDownPayment =(e) =>{
  this.setState({
    downPayment: e.target.value
  });
}
```

```

setDefaultImage (uploadType) {
  if (uploadType === "multer") {
    this.setState({
      multerImage: DefaultImg
    });
  }
}

setImage (e, method) {

  if (method === "multer") {

    this.setState({
      img1:"multer-image-" + Date.now(),
      img2 : e.target.files[0]
    });

    this.setState({
      multerImage: URL.createObjectURL(e.target.files[0])
    });
  }
}

onSubmit =(e) =>{
  e.preventDefault();

  let imageFormObj = new FormData();

  imageFormObj.append("imageName", this.state.img1);
  imageFormObj.append("imageData", this.state.img2);
  imageFormObj.append("serialCode", this.state.serialCode);
  imageFormObj.append("packageName", this.state.packageName);
  imageFormObj.append("packageType", this.state.packageType);
  imageFormObj.append("monthlyCharge", this.state.monthlyCharge);
  imageFormObj.append("downloadSpeed", this.state.downloadSpeed);
  imageFormObj.append("uploadSpeed", this.state.uploadSpeed);
  imageFormObj.append("downloadLimit", this.state.downloadLimit);
  imageFormObj.append("uploadLimit", this.state.uploadLimit);
  imageFormObj.append("extraGBFee", this.state.extraGBFee);
  imageFormObj.append("downPayment", this.state.downPayment);

  axios.post('http://localhost:5000/api/pac/add', imageFormObj)
    .then((data) => {
      if (data.data.success) {
        swal("Successful", "Package Details Added", "success");
        this.props.history.push('/')
      }
    })
    .catch((err) => {
      swal("Unsuccessful", "Package Details Not Added", "error");
    });

  this.setState({
    serialCode: "",
  }

```



```

        <div className="form-group">
            <label>Package Type :</label>
            <div onChange={this.onChangePackageType}>
                <input type="radio" value="4G" name="type"
required/> 4G <br/>
                <input type="radio" value="Fiber"
name="type" required/> Fiber <br/>
                <input type="radio" value="Broadband"
name="type" required/> Broadband
            </div>
        </div>

        <div className="form-group">
            <label>Monthly Rental (Rs) :</label>
            <input
                required
                type="number"
                className="form-control"
                value={this.state.monthlyCharge}
                onChange={this.onChangeMonthlyCharge}
            />
        </div>

        <div className="form-group">
            <label>Download Speed (Mbps) :</label>
            <input
                required
                type="number"
                className="form-control"
                value={this.state.downloadSpeed}
                onChange={this.onChangeDownloadSpeed}
            />
        </div>

        <div className="form-group">
            <label>Upload Speed (Mbps) :</label>
            <input
                required
                type="number"
                className="form-control"
                value={this.state.uploadSpeed}
                onChange={this.onChangeUploadSpeed}
            />
        </div>

        <div className="form-group">
            <label>Download Limit (GB) :</label>
            <input
                required
                type="number"
                className="form-control"
                value={this.state.downloadLimit}
                onChange={this.onChangeDownloadLimit}
            />
        </div>

```

```

        <div className="form-group">
          <label>Upload Limit (GB) :</label>
          <input
            required
            type="number"
            className="form-control"
            value={this.state.uploadLimit}
            onChange={this.onChangeUploadLimit}

          />
        </div>

        <div className="form-group">
          <label>Charge for an Extra GB (Rs) :</label>
          <input
            required
            type="number"
            className="form-control"
            value={this.state.extraGBFee}
            onChange={this.onChangeExtraGBFee}

          />
        </div>

        <div className="form-group">
          <label>Down Payment (Rs) :</label>
          <input
            required
            type="number"
            className="form-control"
            value={this.state.downPayment}
            onChange={this.onChangeDownPayment}

          />
        </div>

        <div className="form-group">
          <label>Add Package Photo :</label>
          <div className="main-container">
            <div className="image-container">
              <div className="process">

                <input required type="file"
className="process__upload-btn" onChange={ (e) => this.setImage(e,
"multer")} />

                <img src={this.state.multerImage}
alt="upload-image" className="process__image" />
              </div>
            </div>
          </div>
        </div>

        <div className="form-group">
          <input type="submit" value="Add Details"
className= "btn btn-primary"/>
        </div>
      </form>

```





```

router.route("/add").post(
  upload.single("imageData"),
  (req, res, next) => {
    console.log(req.body);
    const newPac = new Package({ /*pac cons called*/
      imageName: req.body.imageName,
      // imageData: req.file.path,
      imageData: req.file.path.substr(22),
      serialCode: req.body.serialCode,
      packageName: req.body.packageName,
      packageType: req.body.packageType,
      monthlyCharge: req.body.monthlyCharge,
      downloadSpeed: req.body.downloadSpeed,
      uploadSpeed: req.body.uploadSpeed,
      downloadLimit: req.body.downloadLimit,
      uploadLimit: req.body.uploadLimit,
      extraGBFee: req.body.extraGBFee,
      downPayment: req.body.downPayment,
    });

    newPac
      .save()
      .then((result) => {
        console.log(result);
        res.status(200).json({
          success: true,
          document: result,
        });
      })
      .catch((err) => next(err));
  }
);

```

model/ package.js

```

const mongoose = require('mongoose');

const moment = require('moment');

const packageSchema = mongoose.Schema({

  serialCode: {
    type: String,
    maxlength: 50,
    unique: 1
  },
  packageName : {
    type : String
  },
  packageType: {
    type: String
  },
  monthlyCharge: {
    type : Number
  },

```

```

    downloadSpeed: {
      type : Number
    },
    uploadSpeed: {
      type : Number
    },
    downloadLimit: {
      type : Number
    },
    uploadLimit: {
      type : Number
    },
    extraGBFee: {
      type : Number
    },
    downPayment: {
      type : Number
    },
    imageName: {
      type: String,
      default: "none",
      required: true
    },
    imageData: {
      type: String,
      required: true
    }
  }, {
    collection : 'packages'
  });

const Package = mongoose.model('Packages', packageSchema);
module.exports = Package;

```

## d – View / Search Package List (front end)

searchPackageListComponent.js

```

import React, { Component } from "react";
import { connect } from "react-redux";
import PropTypes from "prop-types";
import { Link } from "react-router-dom";
import Package from "../package.component";
import "react-notifications/lib/notifications.css";
import styles from '../css/searchPackageList.module.css';

class SearchPackageList extends Component {
  state = {

```

```

        pager: {},
        pageOfItems: [],
        query : ""
    };

    static propTypes = {
        isAuthenticated: PropTypes.bool,
        emp: PropTypes.object.isRequired,
    };

    setType = (e) => {
        this.setState({
            query: e,
            pager: {},
            pageOfItems: []
        });
        setTimeout(() => {
            this.loadPage();
        }, 2000)
    }

    setSearch = (e) => {
        this.setState({
            query: e.target.value,
            pager: {},
            pageOfItems: []
        });
    }

    loadPage = () => {
        // get page details and items from api
        const params = new URLSearchParams(window.location.search);
        const page = parseInt(params.get("page")) || 1;
        if (page !== this.state.pager.currentPage && this.state.query !==
        '') {
            fetch(`http://localhost:5000/api/pac/get/all/paginate/search?page=${page}&search=${this.state.query}`, {
                method: "GET",
            })
                .then((response) => response.json())
                .then(({ pager, pageOfItems }) => {
                    this.setState({ pager, pageOfItems });
                });
        }
        this.setState({
            query: ""
        });
    };

    render() {
        const { pager, pageOfItems } = this.state;
        return (
            <div>

                <div style={{height:"50px"}} />
            { /*-----

```

```

-----*/}

    <form onSubmit={this.loadPage}>

        <div className="form-group">
            <label>Advanced Search (Search by Name or
Specialization):</label>
            <input type="text" className="form-control"
value={this.state.query} onChange={this.setSearch}/>

        </div>

        <div className='col-sm-12 row p-0 m-0'>
            <div className='col-sm-6 p-0 m-0 justify-content-
start'>

                <div className="form-group">
                    <input type="submit" value="Search"
className="btn btn-primary"/>
                </div>
            </div>
            <div className='col-sm-6 p-0 m-0 justify-content-
end text-right row'>
                <p className={styles.typeOption} onClick={() =>
this.setType("4G")}>4G</p><h3> | </h3>
                <p className={styles.typeOption} onClick={() =>
this.setType("Fiber")}>Fiber</p><h3> | </h3>
                <p className={styles.typeOption} onClick={() =>
this.setType("Broadband")}>Broadband</p>
            </div>
        </div>

    </form>
<br/>

    { /*-----*/}

-----*/}

    <div className="card text-center mx-0">
        <h3 className="card-header font-weight-bold">Internet
Package List</h3>

        <div className="card-body ">
            {pageOfItems.map((item) => (
                <
                    <Package item={item}/>
                </>
            ))}
        </div>
        <div className="card-footer pb-0 pt-3">
            {pager.pages && pager.pages.length && (
                <ul className="pagination">
                    <li
                        className={`page-item first-item ${
                            pager.currentPage === 1 ?
"disabled" : ""
                        }`}
                    >
                        <Link to={{ search: `?page=1` }}
className="page-link">

```

```

First
</Link>
</li>

<li
  className={`page-item previous-item ${
    pager.currentPage === 1 ?
"disabled" : ""
  }}
>
  <Link
    to={{ search:
`?page=${pager.currentPage - 1}` }}
    className="page-link"
  >
    Previous
  </Link>
</li>

{pager.pages.map((page) => (
  <li
    key={page}
    className={`page-item number-item
      ${
        pager.currentPage === page ?
"active" : ""
      }
    `}
  >
    <Link
      to={{ search: `?page=${page}` }}
      className="page-link"
    >
      {" "}
      {page}{" "}
    </Link>
  </li>
))}
<li
  className={`page-item next-item ${
    pager.currentPage ===
pager.totalPages ? "disabled" : ""
  }}
>
  <Link
    to={{ search:
`?page=${pager.currentPage + 1}` }}
    className="page-link"
  >
    {" "}
    Next{" "}
  </Link>
</li>

<li
  className={`page-item last-item ${
    pager.currentPage ===
pager.totalPages ? "disabled" : ""
  }}
>
  <Link

```

```

                                to={{ search:
`?page=${pager.totalPages}` }}
                                className="page-link"
                                >
                                Last
                                </Link>
                                </li>
                                </ul>
                                )}
                                </div>
                                </div>
                                </div>
                                );
                                }
                                }

const mapStateToProps = (state) => ({
  item: state.item,
  isAuthenticated: state.emp.isAuthenticated,
  emp: state.emp,
});

export default connect(mapStateToProps, null)(SearchPackageList);

```

## package.js

```

import React, { Component } from "react";
import { connect } from "react-redux";
import "react-notifications/lib/notifications.css";

class Package extends Component {

  render() {
    const { item } = this.props;
    return (
      <div key={item._id}>
        <div className="container rounded-0 border border-info ">
          <div className="container ">
            <div className="row">
              <div className="col-sm">
                <br />
                <img
                  alt="packageImg"
                  height="80%"
                  width="100%"
                  src={` /uploads/${item.imageData}`}
                />
                <br />
              </div>
              <div className="col-sm">
                <br />
                <h4 className="font-weight-bold text-left
text-danger mt-4">
                  {item.packageName}
                </h4>
                <h5 className="font-weight-bold text-left
text-info">
                  Monthly Rental : Rs
                  {item.monthlyCharge}.00

```

```

        </h5>
        <h6 className="font-weight-bold text-left">
            Down Payment : Rs {item.downPayment}.00
        </h6>
        <h6 className="font-weight-bold text-left">
            Type : {item.packageType}
        </h6>

    </div>

    <div className="col-sm">
        <br />
        <br />
        <h6 className="font-weight-bold text-left
text-danger mt-3">
            {item.uploadSpeed}Mbps
            Speed :  {item.downloadSpeed}Mbps 
        </h6>
        <h6 className="font-weight-bold text-left
text-primary">
            {item.downloadLimit} GB
            Download Data Limit :
        </h6>
        <h6 className="font-weight-bold text-left
text-success">
            GB
            Upload Data Limit : {item.uploadLimit}
        </h6>
        <p className="font-weight-bold text-left
text-warning">
            {item.extraGBFee}.00
            Charge For Extra GB : Rs
        </p>
    </div>
</div>
</div>
</div>
<br />
</div>

);
}
}

const mapStateToProps = (state) => ({
    isAuthenticated: state.emp.isAuthenticated,
    emp: state.emp,
});

export default connect(mapStateToProps, null)(Package);

```

## View / Search Package List (back end)

route/package.js

```
const paginate = require('jw-paginate');
const express = require('express');
const router = express.Router();
const Package = require('../model/package');
const multer = require("multer");

//package list will be taken without search
router.get('/get/all/paginate', (req,res) => {
    Package.find().then(items => {

        const page = parseInt(req.query.page) || 1;

        // get size of items that should display
        const pageSize = 5;
        const pager = paginate(items.length, page, pageSize);

        // get the page number from item list
        const pageOfItems = items.slice(pager.startIndex, pager.endIndex +
1);

        // return pagination related data and items in the selected page
        return res.json({ pager, pageOfItems });

    });
});

////////////////////////////////////
////////////////////////////////////
//package list will be taken with search
router.get('/get/all/paginate/search', (req,res) => {
    Package.find({
        "$or": [
            { packageName: { '$regex': req.query.sitem, '$options': 'i' } },
            { packageType: { '$regex': req.query.sitem, '$options': 'i' } } ]
        /*to search on package */
    }).then(items => {

        const page = parseInt(req.query.page) || 1;

        // get size of items that should display
        const pageSize = 5;
        const pager = paginate(items.length, page, pageSize);

        // get the page number from item list
        const pageOfItems = items.slice(pager.startIndex, pager.endIndex +
1);

        // return pagination related data and items in the selected page
        return res.json({ pager, pageOfItems });
    });
});
```



```

    });

});

////////////////////////////////////

```

## model/package.js

```

const mongoose = require('mongoose');

const moment = require('moment');

const packageSchema = mongoose.Schema({

  serialCode: {
    type: String,
    maxlength: 50,
    unique: 1
  },
  packageName : {
    type : String
  },
  packageType: {
    type: String
  },
  monthlyCharge: {
    type : Number
  },
  downloadSpeed: {
    type : Number
  },
  uploadSpeed: {
    type : Number
  },
  downloadLimit: {
    type : Number
  },
  uploadLimit: {
    type : Number
  },
  extraGBFee: {
    type : Number
  },
  downPayment: {
    type : Number
  },
  imageName: {
    type: String,
    default: "none",
    required: true
  },
  imageData: {
    type: String,

```

```

        required: true
    }

}, {
    collection : 'packages'
});

const Package = mongoose.model('Packages', packageSchema);
module.exports = Package;

```

### q3. routes/emp.js

```

const express = require('express');
const router = express.Router();
const {User} = require('../model/emp');
const bcrypt = require('bcrypt');
const saltRounds = 10;
const jwt = require('jsonwebtoken');
const auth = require('../middleware/auth');

////////////////////////////////////
////////////////////////////////////
//register emps
router.post('/register', (req,res) =>{

    //To register emp

    const {empUn , empEmail, empPw} = req.body;
    if(!empUn || !empEmail || !empPw){
        return res.status(400).json({msg:'Please Fill All Fields'});
    }

    User.findOne({empUn}).then(user => {
        if(user){
            return res.status(400).json({msg : 'Username Already Exist'})
        }
    })

```

```

    });

    User.findOne({empEmail}).then(user => {
      if(user){
        return res.status(400).json({msg : 'Email Already Exist'})
      }
    });

    const user = new User(req.body);
    /*pw hashing*/
    bcrypt.genSalt(10, (err, salt) => {

      bcrypt.hash(user.empPw, saltRounds, function(error, hash) {
        if (error) {
          throw err;
        }
        user.empPw = hash;

        user.save().then(user => {

          jwt.sign(
            {_id : user._id}, "secret", {expiresIn: 10},
            (error, token) =>{
              if(error) {
                throw error;
              }
              res.json({
                token,
                user: {
                  _id: user._id,
                  empUn: user.empUn,
                  empEmail: user.empEmail
                }
              });
            }
          );
        });
      });

    });

  });
});

////////////////////////////////////
////////////////////////////////////
//login
router.post('/login', (req,res) =>{

  //To login emp

  const {empUn , empPw} = req.body;
  if(!empUn || !empPw){
    return res.status(400).json({msg:'Please Fill All Fields'});
  }
});

```

```

    User.findOne({empUn}).then(user => {
      if(!user){
        return res.status(400).json({msg : 'Invalid Username'})
      }
      /*pw checking, hash passwords are checked here*/
      bcrypt.compare(empPw, user.empPw).then(result => {
        if(!result){
          return res.status(400).json({
            msg:'Invalid Credentials'
          });
        }

        jwt.sign(
          {_id : user._id}, "secret", {expiresIn: 3500},
          (error, token) =>{
            if(error) {
              throw error;
            }
            res.json({
              token,
              user: {
                _id: user._id,
                empUn: user.empUn,
                empEmail: user.empEmail
              }
            });
          }
        );
      });
    });

  });

});

////////////////////////////////////
////////////////////////////////////

router.get('/get/emp', auth, (req, res) => {
  User.findById(req.user._id).select('-empPw').then(user => res.json(user));
});

module.exports = router;

```

q4.

```
Package.find({
  "$or": [
    { packageName: { '$regex': req.query.sitem, '$options': 'i' } },
    { packageType: { '$regex': req.query.sitem, '$options': 'i' } }
  /*to search package */
  ]
})
```

Related Model –

```
const mongoose = require('mongoose');
const moment = require('moment');

const packageSchema = mongoose.Schema({

  serialCode: {
    type: String,
    maxlength: 50,
    unique: 1
  },
  packageName : {
    type : String
  },
  packageType: {
    type: String
  },
  monthlyCharge: {
    type : Number
  },
  downloadSpeed: {
    type : Number
  },
  uploadSpeed: {
    type : Number
  },
  downloadLimit: {
    type : Number
  },
  uploadLimit: {
    type : Number
  },
  extraGBFee: {
    type : Number
  },
  downPayment: {
    type : Number
  },
  imageName: {
    type: String,
    default: "none",
    required: true
  },
  imageData: {
    type: String,
    required: true
  }
});
```

```

    }

  }, {
    collection : 'packages'
  });

const Package = mongoose.model('Packages', packageSchema);
module.exports = Package;

```

q5.

packageList.test.js

```

import React from 'react';
import { shallow } from 'enzyme';
import PackageList from "../components/packageList.component";

test('should render PackageList Component correctly', () => {
  const wrapper = shallow(<PackageList />);
  expect(wrapper).toMatchSnapshot();
});

test('should render PackageList Component with Carousel', () => {
  const wrapper = shallow(<PackageList />);
  expect(wrapper.find('Carousel')).toHaveLength(1);
});

test('should render PackageList Component without Add Package functionality', () => {
  const wrapper = shallow(<PackageList />);
  expect(wrapper.find('AddPackage')).toHaveLength(0);
});

test('should render PackageList Component with display package functionality', () => {
  const wrapper = shallow(<PackageList />);
  expect(wrapper.find('Package')).toHaveLength(1);
});

```

## searchPackageList.test.js

```
import React from 'react';
import { shallow } from 'enzyme';
import SearchPackageList from "../components/searchPackageList.component";

test('should render SearchPackageList Component correctly', () => {
  const wrapper = shallow(<SearchPackageList />);
  expect(wrapper).toMatchSnapshot();
});

test('should render SearchPackageList Component without Carousel', () => {
  const wrapper = shallow(<SearchPackageList />);
  expect(wrapper.find('Carousel')).toHaveLength(0);
});

test('should render SearchPackageList Component without Add Package functionality', () => {
  const wrapper = shallow(<SearchPackageList />);
  expect(wrapper.find('AddPackage')).toHaveLength(0);
});

test('should render SearchPackageList Component with display package functionality', () => {
  const wrapper = shallow(<SearchPackageList />);
  expect(wrapper.find('Package')).toHaveLength(1);
});
```





## q6. HomePage

← → ↻ 📄 localhost:3000

☆ ⚙️ 🔍

CellTel Telecommunications

Employee




UNLIMITED  
for a month  
Reload Rs. 286/-

TTC/AB/PRQ/20/06

Advanced Search

Internet Package List





Heavy Gamer

Monthly Rental : Rs 4000.00

Down Payment : Rs 5000.00


Type : Fiber

Speed :  100Mbps  50Mbps

Download Data Limit : 150 GB

Upload Data Limit : 90 GB

Charge For Extra GB : Rs 100.00





Family Plus

Monthly Rental : Rs 2000.00

Down Payment : Rs 1000.00


Type : 4G

Speed :  20Mbps  10Mbps

Download Data Limit : 70 GB

Upload Data Limit : 50 GB

Charge For Extra GB : Rs 200.00





Office Lite

Monthly Rental : Rs 2500.00

Down Payment : Rs 2000.00


Type : 4G

Speed :  20Mbps  10Mbps

Download Data Limit : 100 GB

Upload Data Limit : 50 GB

Charge For Extra GB : Rs 150.00





Office Pal

Monthly Rental : Rs 5000.00

Down Payment : Rs 5000.00


Type : Fiber

Speed :  100Mbps  50Mbps

Download Data Limit : 200 GB

Upload Data Limit : 100 GB

Charge For Extra GB : Rs 50.00





Office Pro

Monthly Rental : Rs 25000.00

Down Payment : Rs 10000.00

Type : Fiber

Speed :  150Mbps  100Mbps

Download Data Limit : 500 GB

Upload Data Limit : 250 GB

Charge For Extra GB : Rs 50.00

First Previous 1 2 Next Last

CellTel Telecommunication Network is One of the  
Leading Internet Services Providers in Sri Lanka  
That Provide Their Customers With Quality  
Services.

- CellTel Hub, Colombo, Sri Lanka
- +94 912 222 199
- info@celtel.lk


© CellTel. All rights reserved

-CellTel (PVT) Ltd-  
FUTURE ALWAYS

CellTel Main Hub

Yatagala Raja Mah...

This map was created by a user.



Map data ©2021 Terms 307M



## Search

← → ↻ 📄 localhost:3000/search ☆ ⚙️ N ⋮

CelTel Telecommunications 


Employee

Advanced Search (Search by Name or Specialization):  
  

Search

4G|Fiber|Broadband

Internet Package List



Heavy Gamer

Monthly Rental : Rs 4000.00

Down Payment : Rs 5000.00

Type : Fiber

Speed : 


100Mbps

50Mbps

Download Data Limit : 150 GB

Upload Data Limit : 90 GB

Charge For Extra GB : Rs 100.00



Office Pal

Speed : 

100Mbps

50Mbps

## Employee Sign In page

The screenshot shows a web browser at localhost:3000/loginEmp. The page has a dark header with 'CelTel Telecommunications' and an 'Employee' button. A red 'Employee Register' button is in the top right. The main content area is titled 'Employee Sign In' and contains two input fields for 'Username :' and 'Password :', followed by a blue 'Login' button. The footer includes a company description, the logo '-CelTel (PVT) Ltd- FUTURE ALWAYS', and a 'CelTel Main Hub' section with a map of 'Yatagala Raja Mah...'.

CelTel Telecommunications

Employee

Employee Register

### Employee Sign In

Username :

Password :

Login

CelTel Telecommunication Network is One of the Leading Internet Services Providers in Sri Lanka That Provide Their Customers With Quality Services.

-CelTel (PVT) Ltd-  
FUTURE ALWAYS

CelTel Main Hub

Yatagala Raja Mah...  
This map was created by a user.

## Employee Register Page

The screenshot shows a web browser at localhost:3000/registerEmp. The page has a dark header with 'CelTel Telecommunications' and an 'Employee' button. The main content area is titled 'Employee Sign Up' and contains five input fields for 'Username :', 'Email Address :', 'Password :', 'Confirm Password :', and 'PIN (To Confirm as a Real Employee) :', followed by a blue 'Register' button. The footer includes a company description, contact information, the logo '-CelTel (PVT) Ltd- FUTURE ALWAYS', and a 'CelTel Main Hub' section with a map of 'Yatagala Raja Mah...'.

CelTel Telecommunications

Employee

### Employee Sign Up

Username :

Email Address :

Password :

Confirm Password :

PIN (To Confirm as a Real Employee) :

Register

CelTel Telecommunication Network is One of the Leading Internet Services Providers in Sri Lanka That Provide Their Customers With Quality Services.

- CelTel Hub, Colombo, Sri Lanka
- +94 912 222 199
- info@celtel.lk

© CelTel. All rights reserved

-CelTel (PVT) Ltd-  
FUTURE ALWAYS

CelTel Main Hub

Yatagala Raja Mah...  
This map was created by a user.

Google My Maps

Map data ©2021 Terms 50 m

## Add Package Details Page

← → ↻ 📄 localhost:3000/addPackage#!

☆ 🌐 N ⋮

CellTel Telecommunications

Hi emp1Add PackageLogout

### Add Package Details

Package Serial Code :

Package Name :

Package Type :

☐ 4G  
☐ Fiber  
☐ Broadband

Monthly Rental (Rs) :

Download Speed (Mbps) :

Upload Speed (Mbps) :

Download Limit (GB) :


Upload Limit (GB) :

Charge for an Extra GB (Rs) :

Down Payment (Rs) :

Add Package Photo :

Choose File No file chosen



Add Details

CellTel Telecommunication Network is One of the  
Leading Internet Services Providers in Sri Lanka  
That Provide Their Customers With Quality  
Services.


- CellTel Hub, Colombo, Sri Lanka
- +94 912 222 199
- info@celtel.lk

© CellTel. All rights reserved

-CellTel (PVT) Ltd-  
FUTURE ALWAYS

#### CellTel Main Hub

Yatagala Raja Mah...  
This map was created by a user.



Map data ©2023 © 2023 © 2023 © 2023