



Rajiv Gandhi University of Knowledge Technologies

(Catering the Educational Needs of Gifted Rural Youth of A.P)

R.K Valley, Y.S.R Kadapa (Dist)-516330

OUTPASS MANAGEMENT SYSTEM

Supervised by

Mrs.SUSMITHA E

Assistant Professor

Department of Computer Science Engineering

Team Members:

1. D.Haseena - R170700
2. S.Charitha - R170690

This Project report has been submitted in fulfilment of the requirements for the Degree of Bachelor of Technology in Software Engineering.

CERTIFICATE

This is to certify that the project work titled “OUTPASS MANAGEMENT SYSTEM” is a bonafied project work submitted by D.Haseena And S.Charitha in the department of COMPUTER SCIENCE AND ENGINEERING in partial fulfilment of requirements for the award of degree of Bachelor of Technology in Computer science and engineering for the year 2022-2023 carried out the work under the supervision.

Project Supervisor

Mrs. SUSMITHA E
Assistant professor
Dept Of CSE

HEAD OF THE DEPARTMENT

Mr. N. SATYANANDARAM
Lecturer.
Dept Of CSE

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be complete without the mention of the people who made it possible and whose constant guidance and encouragement crown all the efforts success.

I am extremely grateful to our respected Director Prof. K. SANDHYA RANI for fostering excellent academic climate in our institution.

I also express my sincere gratitude to our respected Head of the Department Mr.SATYANANDARAM for his encouragement, overall guidance in viewing this project a good asset and effort in bringing out this project. I would like to convey thanks to our guide at college Mrs.Susmitha E for her guidance, encouragement, co-operation and kindness during the entire duration of the course and academics.

My sincere thanks to all the members who helped me directly and indirectly in the completion of project work. I express my profound gratitude to all our friends and family members for their encouragement.

INDEX

S.NO	INDEX	PAGE NUMBER
1	Abstract	1
2	Introduction	1
3	Purpose	2-3
4	Scope	4-5
5	Requirement Specification	5-7
6	Analysis and Design	7-10
6.1	Use case Diagram	11-13
6.2	ER Diagrams	13-16
7	Implementation and System Testing	17-21
8	Project Output	22-29
9	Conclusion	30

ABSTRACT

Packages is a web application which shows categories of adventurous places. Under this categories, a list of places will be displayed. Anyone who visits the website can check the packages. With the help of instructions they can make quick booking for which they want to travel. The places which we are displayed are very adventurous and enthusiastic.

INTRODUCTION

In today's running technology every job has become effortless, keeping this in mind a system has been designed which is helpful for colleges, universities & other educational institutes. This is designed for issuing & maintaining the out pass for students departing outside the campus assuring that they have departed with intimation to their Wardens responsible authorities. There are as many systems, which are being used for this purpose, but they work in native nature. To avoid the demerits of existing system this system is designed intelligently to serve numerous users at an instant. It is purely network-based miniature. A high-level security Issue to avoid anonymous entry to system, and to avoid worthless tapping of data

Purpose

Outpass system for hostels where students get to take permission of Warden/care taker just by filling up a simple form. The student login's and fills the out-pass form and waits for the reply from the authorities(caretaker/warden). By default, the status of the out-pass is "pending". The Warden gets all the Outpass Applications from students and have the power whether to accept or reject the out-pass. Warden/care taker can take further steps of verification and decides whether to approve or reject the Out-pass. If accepted the status is "Approved" or if rejected the status is "rejected" and the status displayed on the student's check

status page. Warden can also able to see the all approved and rejected out passes details.

Intended Audience

The intended audience for this document includes the interested software developers and the travel agents who are going to use this software. This project is being developed under the guidance of Mrs.Susmitha E. This software is also useful for the customers in directly accessing the website.

MODULES that can be included in the OUTPASS management system application SRS are as follows:

Home Page – basic details of particular student should be shown.

Apply – Students can fill the application form for the out-pass. Here Student must fill the basic details like Guardian Name, Guardian Ph. No., Student Ph. No., From Date, To Date, Reason for Out-pass.

Check Status – Students can check the status of his/her out passes.

Reset Password

Logout

Warden must have following functional requirements

Home – Basic details of warden should be shown

Pending List - Details of all out-passes of status as 'pending' should be shown.

Accepted List - Details of all out-passes of status as 'approved' should be shown.

Rejected List - Details of all out-passes of status as 'rejected' should be shown.

Reset Password

Logout

Project Scope

This Online Employee Outpass Management System in Node JS MongoDB project is primarily concerned with keeping track of workers' leave activities. To be more specific, the system aids in the tracking of personnel, departments, and leave records according to available categories. In addition, the system shows all of the workers' leave histories. In addition, the system provides for the management of leave kinds and other features. Obviously, this project has an admin panel as well as an employee panel. An employee has a modest position and power over the system in this web application's overview. He or she may ask for leave and keep track of their own absences. In order to request for leave days, an employee must state specific things. Such is the sort of leave, the conditions, and the start and end dates. Aside from that, an employee may change their password and edit their profile.

Intended Audience and Reading Suggestions

The Software Requirements document is intended for:

- Developers who can review a project's capabilities and more easily understand where their efforts should be targeted to improve or add more features to it (designed code the application-it sets the guidelines for future development).
- Project testers can use this document as a base for their testing strategy as some bugs are easier to find using a requirements document.
- End users of this application who wish to know about the available places that are present in the website.

OVERALL DESCRIPTION

Product Functions

- Shows available modules
- Applying for leave

Operating Environment

- All Operating Systems
- Some of the browsers **Software**

Requirements

Scripting Languages

- HTML
- CSS
- Node JS
- JAVA SCRIPT
- MangoDB

Hardware Requirements

A computer system or laptop with basic configuration.

SOFTWARE REQUIREMENT SPECIFICATION

Software Requirement Specification is a description of full software system requirements. Software Requirement Specification describes the behaviour of software from user's point of the view The functional requirements and non-functional requirements are following:

Functional Requirements

1. User Registration

This module covers the details of the users which are required for the registration. User can register itself by adding data like name, password,

email id and further details. After registration they can be sign in by their username and password.

2. Leave Booking

This module maintain the Leave booking For Students who are applying the leave, How many days of leave does he/she want, What is the Reason for the Leave etc..

Non-Functional Requirements

1.Reliability

User should get appropriate information about students and wardens and leave Tracking. Sensitive details of users are safe and secured.

2.Usability

Checking that the system is easy to handle and navigates in the most expected ways with no delays. It has good graphical user interface. It is user friendly. So user feel easy to use.

3.Availability

User should get information 24x7. User can access this software anytime, anywhere Most of the requests sent to the application should be answered within less time.

4.Performance

Outpass Management System application should be able to respond to the Request submitted by the Student without much delay.

5.Security

As it is a web based application it should be more secure in order to save confidential data from hackers. Our software hides the confidential data like contact info of our users.

6.Platform Compatibility

This tool will work on any kind of operating system without modifying it.

7.Efficiency

It is efficient for all user Because it is easy to use and easy to understand. It has simple way of work that user want to do.

Software Requirement Analysis

Software requirement analysis is important part of our project. If requirement of project is clear then a project can be done easily. Our objectives for software requirement are:

Objective/Goals

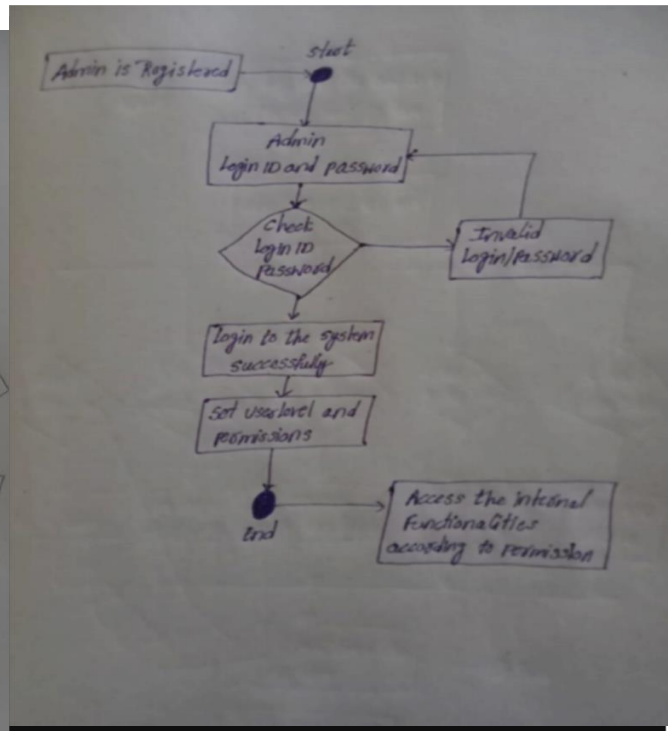
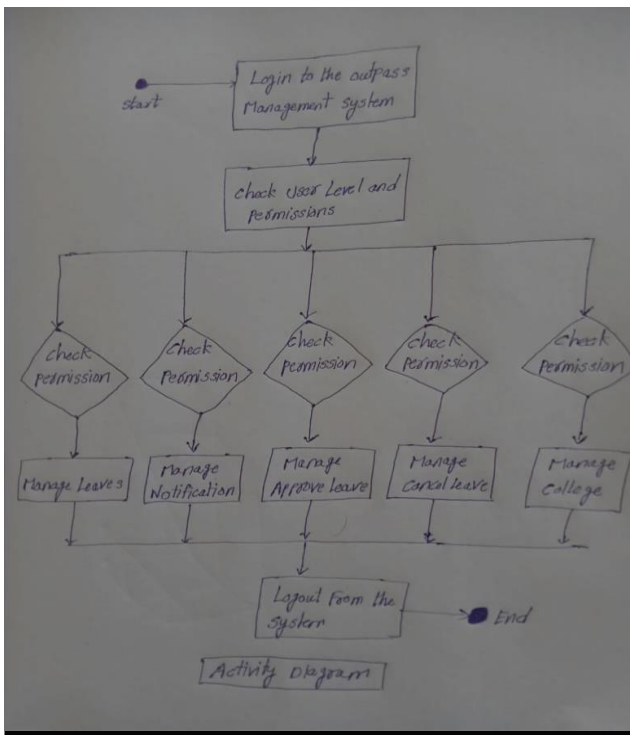
- Students can submit their Leave Request through online.
- Admin can know which problem and where and can know everything about website.

UML and System Diagram

UML stands for Unified Modelling Language. UML is a language for specifying, visualizing and documenting the system. This is the step while developing any product after analysis. The goal from this is to produce a model of the entities involved in the project which later need to be built. The representation of the entities that are to be used in the product being developed need to be designed.

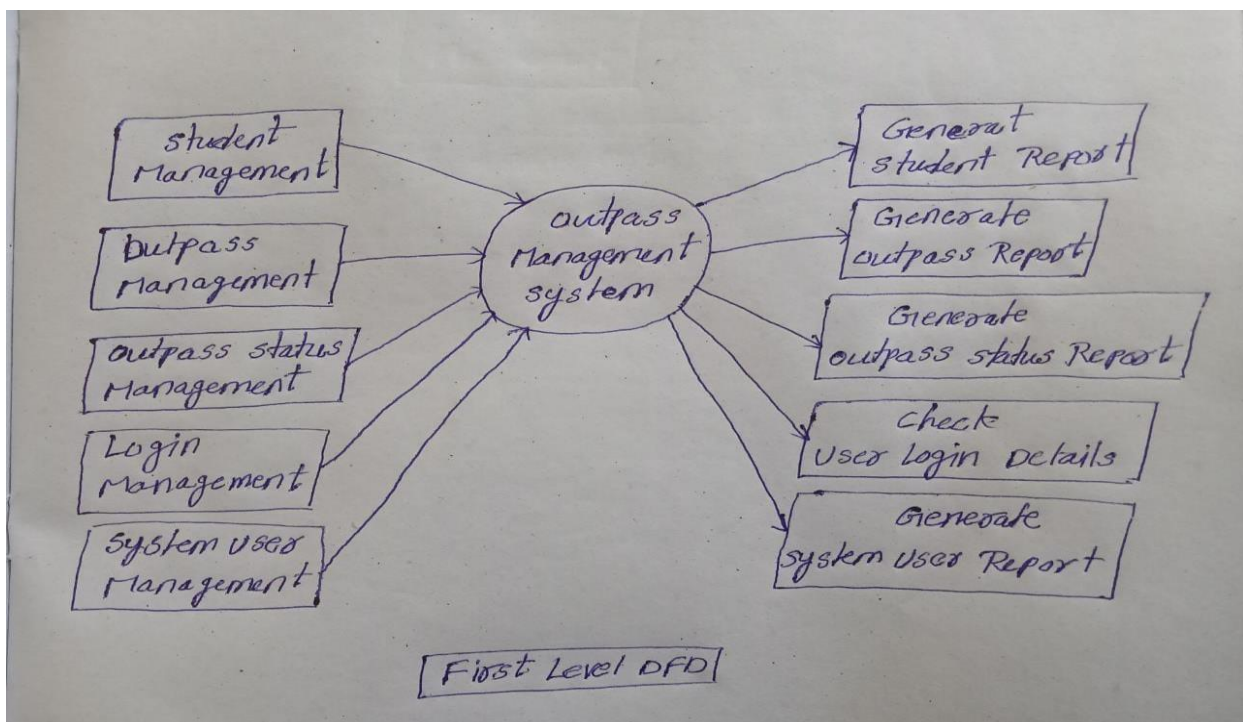
Design is the first step in the development phase for any techniques and principles for the purpose of defining a device, a process or system in sufficient detail to permit its physical realization. Once the software requirements have been analyzed and specified the software design involves three technical activities - design, coding, implementation and testing that are required to build and verify the software.

Activity Diagram

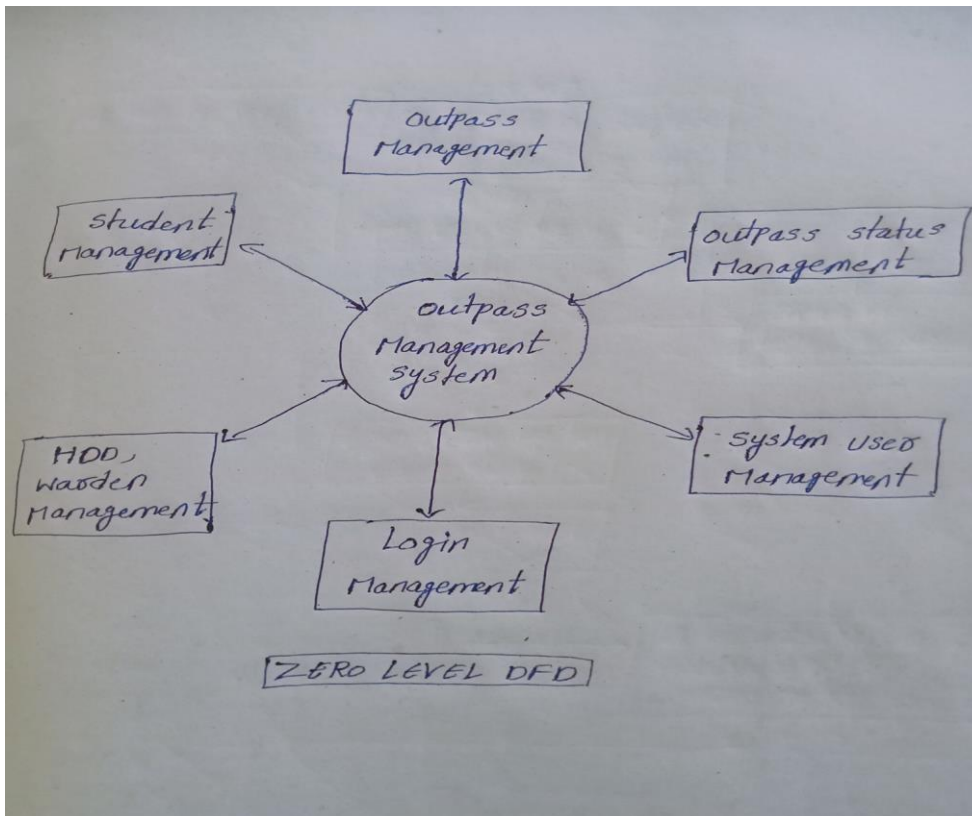


DFD Diagrams

First Level DFD



Zero level DFD

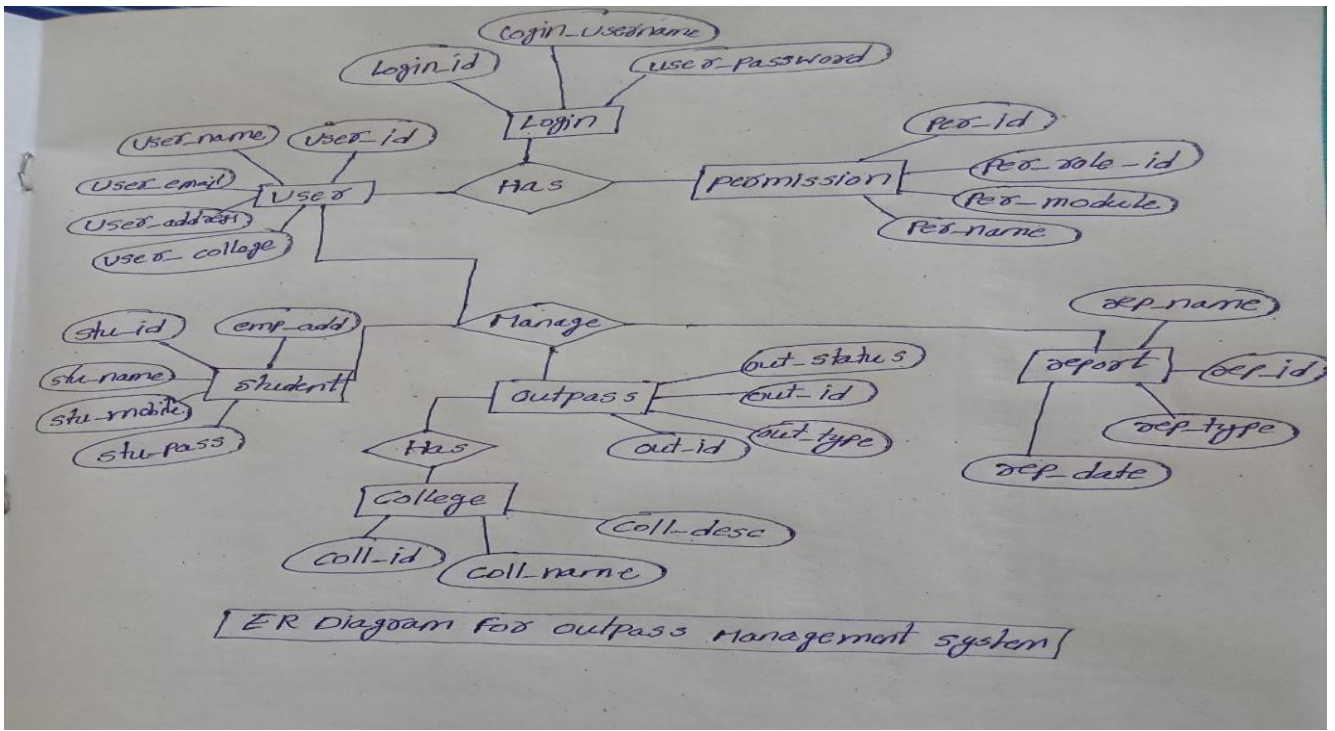


ER-Diagram

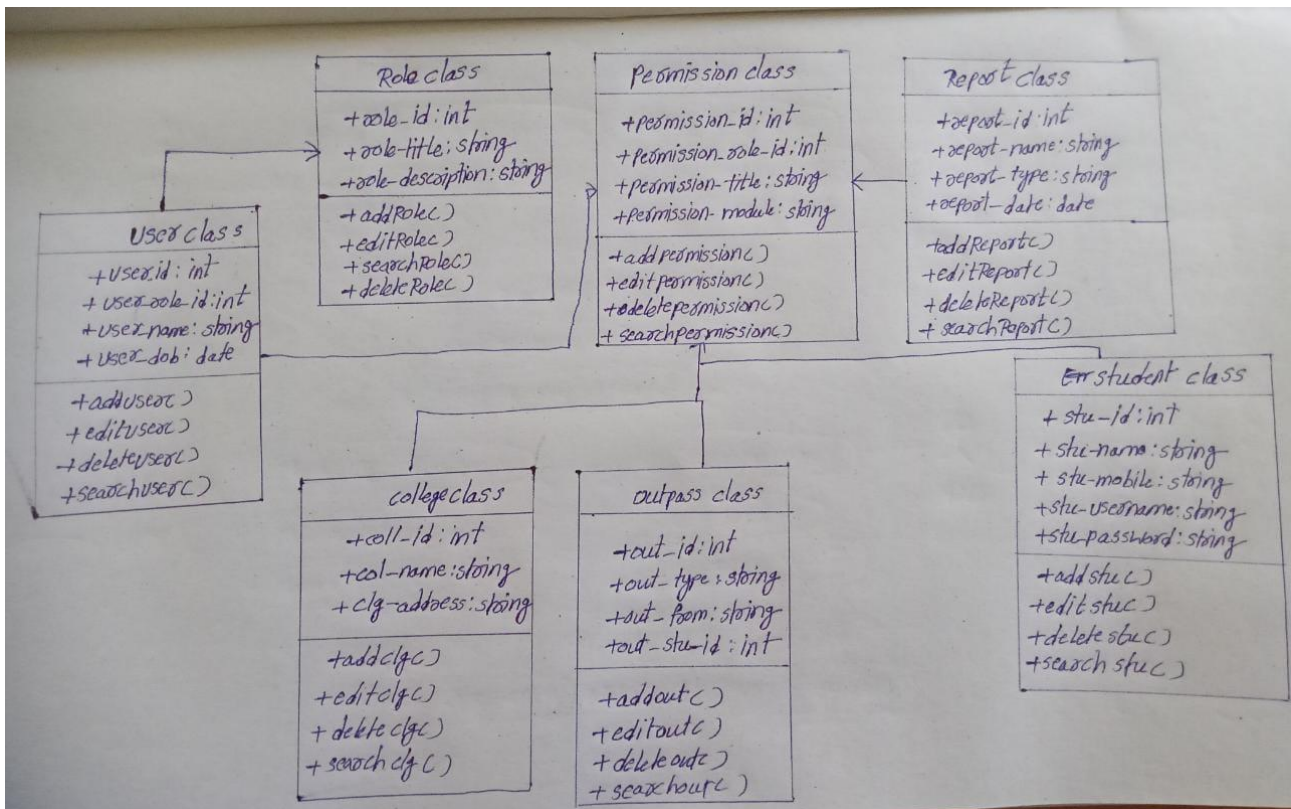
The Entity-Relationship (ER) model was originally proposed by Peter in 1976 [Chen76] as a way to unify the network and relational database views. Simply stated the ER model is a conceptual data model that views the real world as entities and relationships. A basic component of the model is the Entity Relationship diagram which is used to visually represent data objects. Since wrote his paper the model has been extended and today it is commonly used for database design for the database designer, the utility of the ER model is: ->It maps well to the relational model. The constructs used in the ER model can easily be transformed into relational tables.

->It is simple and easy to understand with a minimum of training. Therefore, the model can be used by the database designer to communicate the design to the end user.

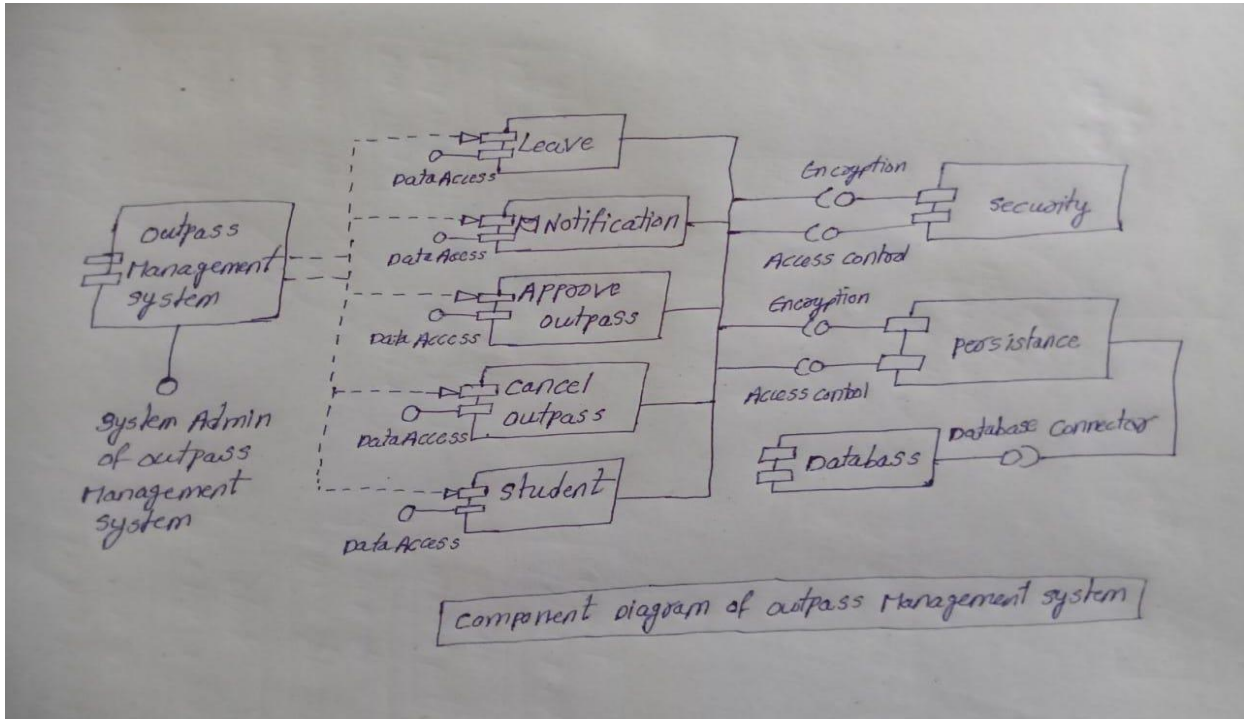
->In addition, the model can be used as a design plan by the database developer to implement a data model in specific database management software.



Class Diagram

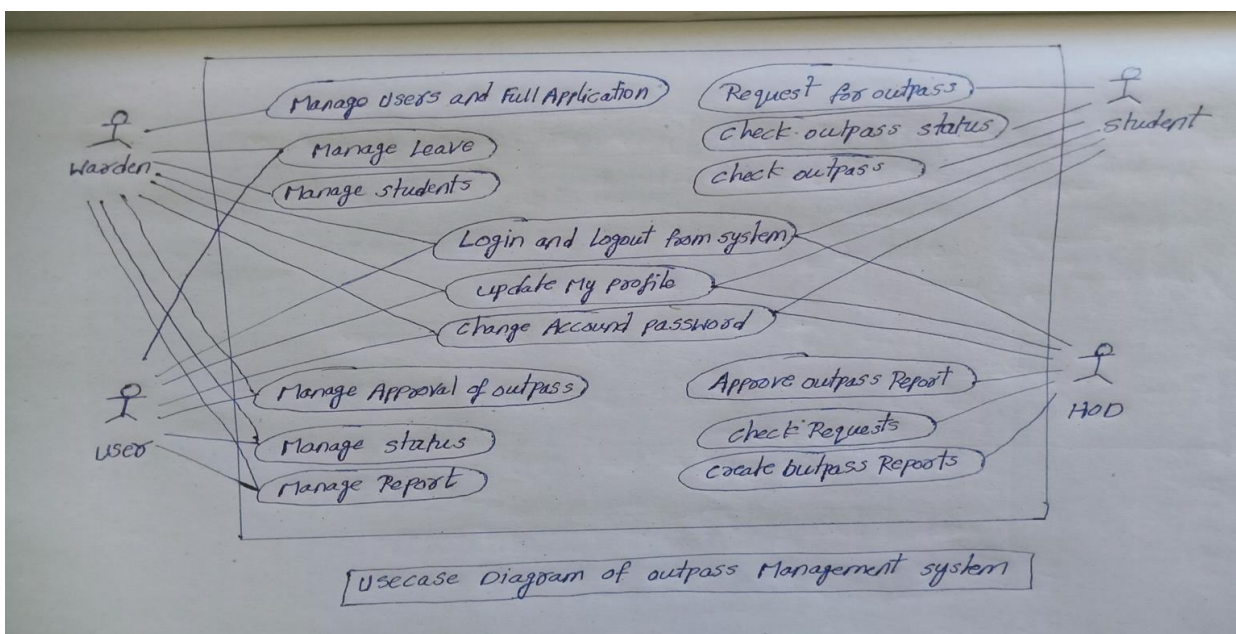


Component Diagram

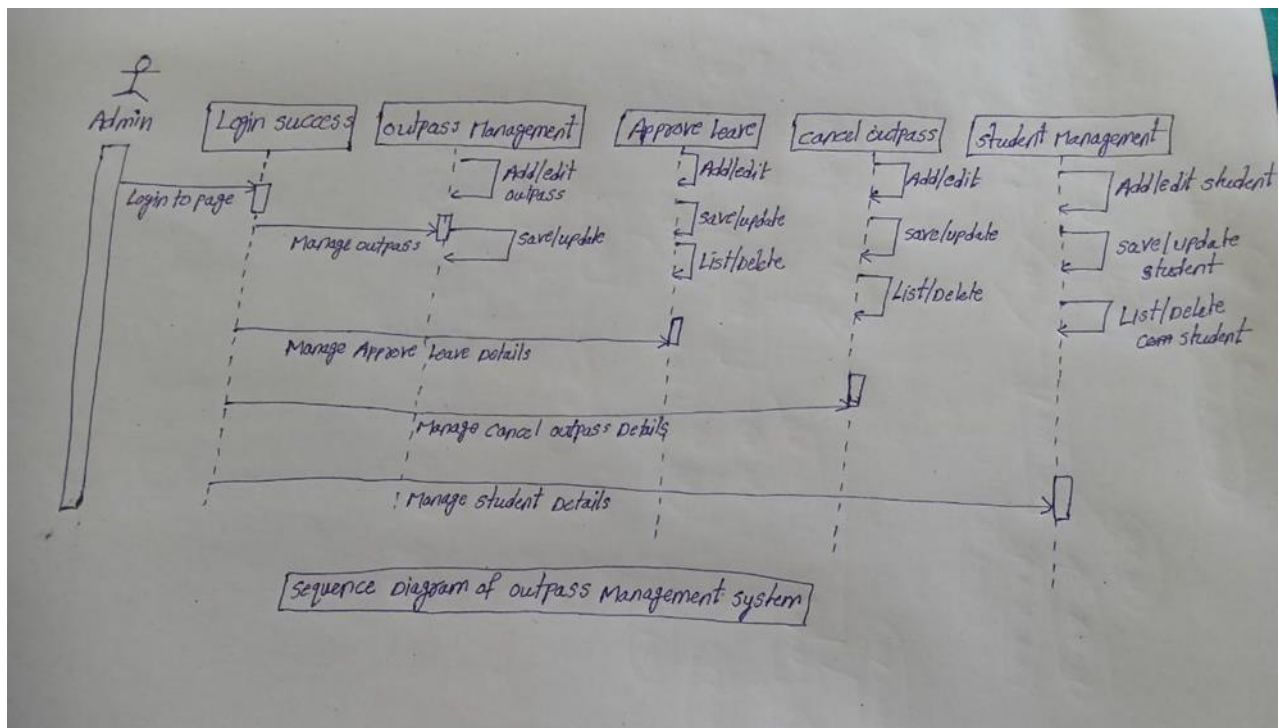


Use Case Diagram

A Use case is a description of set of sequence of actions. Graphically it is rendered as an ellipse with solid line including only its name. Use case diagram is a behavioral diagram that shows a set of use cases and actors and their relationship. It is an association between the use cases and actors. An actor represents a real-world object. Primary Actor – Sender, Secondary Actor Receiver.



Sequence Diagram



Feature Available Outpass Management System using Node JS and MongoDB

- Student Panel
- Warden Panel
- Apply for Leave
- History of Leave
- Managing Profile
- HOD panel

Implementation and System Testing

After all phase have been perfectly done, the system will be implemented to the server and the system can be used.

System Testing

The goal of the system testing process was to determine all faults in our project .The program was subjected to a set of test inputs and many explanations were made and based on these explanations it will be decided whether the program behaves as expected or not. Our Project went through two levels of testing

1. Unit testing

2 .Integration testing

Unit Testing

Unit testing is commenced when a unit has been created and effectively reviewed .In order to test a single module we need to provide a complete environment i.e. besides the section we would require The procedures belonging to other units that the unit under test calls Non local data structures that module accesses .A procedure to call the functions of the unit under test with appropriate parameters

1. Test for the admin module

Testing admin login form-This form is used for log in of administrator of the system. In this

form we enter the username and password if both are correct administration page will open

otherwise if any of data is wrong it will get redirected back to the login page and again ask the details.

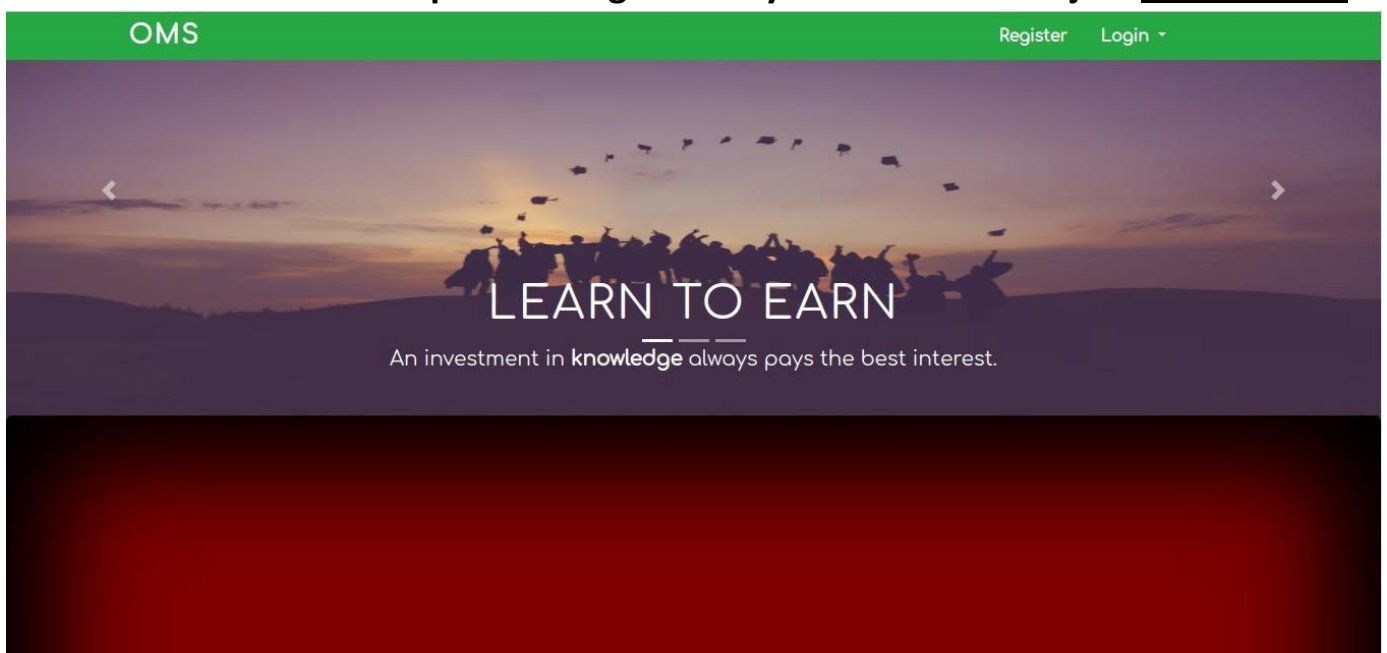
Report Generation: admin can generate report from the main database.

Integration Testing

In the Integration testing we test various combination of the project module by providing the input.

The primary objective is to test the module interfaces in order to confirm that no errors are occurring when one module invokes the other module.

Screenshots Features Outpass Management System Node JS Project HOME PAGE



Registration and Login Form

OMS

RegisterLogin

Register

Name

name

Usertype

☐ Student ☐ HOD ☐ Warden

Username

username

Password

password

Confirm Password

Confirm password

Department

Department

Hostel

Hostel

Image Url

img url

Submit

Back

STUDENT PANEL

OMS

Logout


Welcome, nisha

Home

Profile

Apply For Leave

Track Leave



Name

: nisha

Type

: student

Username

: nisha

Department

: Computer Science

Hostel

: kundhu

APPLY FOR LEAVE

Oms

Logout

Welcome, nisha

Home

Profile

Apply For Leave

Track Leave

Subject	From	To	Days	HOD Status	Warden Status	Final Status
home	14/02/2023	16/02/2023	2	pending	pending	pending

WARDEN PANEL

Oms

RegisterLogin

Warden Login

Username

password

Login

Don't have account ? Register

HOD PANEL

OMS

Register Login ▾

HOD Login

Login

Don't have account ? [Register](#)

Code for Outpass Management System

App.js

```
var express = require("express"),
app = express(),
mongoose = require("mongoose"),
expressvalidator = require("express-validator"),
session = require("express-session"),
methodOverride = require("method-override"),
bodyparser = require("body-parser"), passport
= require("passport"),
LocalStrategy = require("passport-local").Strategy,
passportLocalMongoose = require("passport-local-mongoose"),
flash = require("connect-flash"), Student =
require("./models/student"), Warden =
require("./models/warden"),
Hod = require("./models/hod"), Leave = require("./models/leave"); var
moment = require("moment"); var url = process.env.DATABASEURL ||
"mongodb://localhost:27017/LeaveApp"; mongoose
.connect(url, {
useNewUrlParser: true,
useCreateIndex: true,
```

```

useUnifiedTopology: true,
useFindAndModify: false
})
.then(() => {
  console.log("connected to DB");
})
.catch(err => {
  console.log("Error:", err.message);
});
app.set("view engine", "ejs"); app.use(methodOverride("_method"));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(express.static(__dirname + "/public")); app.use(expressvalidator());
app.use(passport.initialize()); app.use(passport.session()); function
ensureAuthenticated(req, res, next) {
  if (req.isAuthenticated()) {
return next();
  } else {
    req.flash("error", "You need to be logged in");
res.redirect("/student/login");
  }
}
//registration logic
app.post("/student/register", (req, res) => {
var type = req.body.type; if (type ==
"student") { var name = req.body.name;
var username = req.body.username; var
password = req.body.password; var
password2 = req.body.password2; var
hostel = req.body.hostel; var
department = req.body.department; var
image = req.body.image;
  //validation
  req.checkBody("name", "name is required").notEmpty();
req.checkBody("username", "Username is required").notEmpty();
req.checkBody("hostel", "hostel is required").notEmpty();
req.checkBody("department", "department is required").notEmpty();
req.checkBody("password", "Password is required").notEmpty(); if
(errors) { res.render("register", { errors: errors
  });

```

```

    } else {
        var newHod = new Hod({
name: name,      username:
username,      password:
password,      department:
department,      type: type,
image: image
        });
        Hod.createHod(newHod, (err, hod) => {
if (err) throw err;
        console.log(hod);
        });
app.post("/student/register", (req, res) => {
var type = req.body.type; if (type ==
"student") { var name = req.body.name;
var username = req.body.username; var
password = req.body.password; var
password2 = req.body.password2; var
hostel = req.body.hostel; var
department = req.body.department; var
image = req.body.image;
    //validation
    req.checkBody("name", "name is required").notEmpty();
req.checkBody("username", "Username is required").notEmpty();
req.checkBody("hostel", "hostel is required").notEmpty();
req.checkBody("department", "department is required").notEmpty();
req.checkBody("password", "Password is required").notEmpty();
    req.checkBody("password2", "Password dont match").equals(req.body.password);
var errors = req.validationErrors(); if (errors) {
    // req.session.errors = errors;
// req.session.success = false;
console.log("errors: " + errors);
res.render("register", {
    errors: errors
    });
} else {
    var newStudent = new Student({
        name: name,
username: username,
password: password,

```

```

        department: department,
        hostel: hostel,
type: type,
        image: image
    });
    Student.createStudent(newStudent, (err, student) => {
if (err) throw err;    console.log(student);
    });
    // req.flash("success", "you are registered successfully,now you can login");
res.redirect("/student/login");
    }
    } else if (type == "hod") {    var name =
req.body.name;    var username =
req.body.username;    var password =
req.body.password;    var password2 =
req.body.password2;    var department =
req.body.department;    var image =
req.body.image;
    req.checkBody("name", "Name is required").notEmpty();
req.checkBody("username", "Username is required").notEmpty();
req.checkBody("password", "password is required").notEmpty();
req.checkBody("department", "department is required").notEmpty();
req.checkBody("password2", "Password dont match").equals(req.body.password);
var errors = req.validationErrors();    if (errors) {
    res.render("register", {
        errors: errors
    });
    } else {
        var newHod = new Hod({
            name: name,
username: username,
password: password,
department: department,
type: type,
            image: image
        });
        Hod.createHod(newHod, (err, hod) => {
if (err) throw err;
            console.log(hod);
        });
    }
}

```

```

    // req.flash("success", "you are registered successfully,now you can login");
    res.redirect("/hod/login");
  }
} if
(errors) {
  res.render("register", {
errors: errors
  });
} else {
  var newWarden = new Warden({
    name: name,
username: username,
password: password,
hostel: hostel,    type:
type,    image: image
  });
  Warden.createWarden(newWarden, (err, warden) => {
if (err) throw err;    console.log(warden);
  });
  // req.flash("success", "you are registered successfully,now you can login");
  res.redirect("/warden/login");
  }
} });
app.get("/student/:id", ensureAuthenticated, (req, res) => {
  console.log(req.params.id);
  Student.findById(req.params.id)
    .populate("leaves")
    .exec((err, foundStudent) => {
if (err || !foundStudent) {
    req.flash("error", "Student not found");
    res.redirect("back");
  } else {
    res.render("profilestud", { student: foundStudent });
  }
});
});
app.put("/student/:id", ensureAuthenticated, (req, res) => {
  console.log(req.body.student);
  Student.findByIdAndUpdate( req.params.id,

```

```

    req.body.student, (err,
updatedStudent) => {
    if (err) {
        req.flash("error", err.message);
res.redirect("back");
    } else {
        req.flash("success", "Succesfully updated");
        res.redirect("/student/" + req.params.id);
    }
}
});
app.post("/student/:id/apply", (req, res) => {
    Student.findById(req.params.id)
        .populate("leaves")
        .exec((err, student) => {
            if (err) {
                res.redirect("/student/home");
            } else {
                date = new Date(req.body.leave.from);
                todate = new Date(req.body.leave.to);
                year = date.getFullYear();    month =
                date.getMonth() + 1;    dt =
                date.getDate();
                todt = todate.getDate();
                if (dt < 10) {    dt = "0" +
                dt;
            }
            if (month < 10) {
                month = "0" + month;
            }
            console.log(todt - dt);

            Leave.create(req.body.leave, (err, newLeave) => {
                if (err) {
                    req.flash("error", "Something went wrong");
                    res.redirect("back");
                } else {
                    console.log(err);
                }
            });
        });
    });
}
});

```



```

        newLeave.stud.id = req.user._id;        newLeave.stud.username =
req.user.username;        console.log("leave is applied by--" + req.user.username);
        // console.log(newLeave.from);
        newLeave.save();
student.leaves.push(newLeave);
        student.save();
        req.flash("success", "Successfully applied for leave");
        res.render("homestud", { student: student, moment: moment });
    }
    });
}
});
});
);
app.get("/hod/:id/leave", (req, res) => {
    Hod.findById(req.params.id).exec((err, hodFound) => {
    if (err) {
        req.flash("error", "hod not found with requested id");
res.redirect("back");
    } else {
        // console.log(hodFound);
        Student.find({ department: hodFound.department })
        .populate("leaves")
        .exec((err, students) => {
            if (err) {
                req.flash("error", "student not found with your department");
res.redirect("back");
            } else {
app.post("/warden/:id/leave/:stud_id/info", (req, res) => {
Warden.findById(req.params.id).exec((err, wardenFound) => {
if (err) {
    req.flash("error", "warden not found with requested id");
res.redirect("back");
    } else {
//logout for student app.get("/logout",
(req, res) => { req.logout();
    // req.flash("success", "you are logged out");
    res.redirect("/");
});
const port = process.env.PORT || 3005;

```

```
app.listen(port, () => {  
  console.log(`Server started at port ${port}`);  
});
```

Conclusion

This Online Student Outpass Management System in Node JS MongoDB project is primarily concerned with keeping track of student outpass activities. The system aids in the tracking of personal, departments, and outpass records according to available categories. In order to request for outpass days, a student must state specific things such as the sort of leave, the conditions, and the start and end dates.

Warden and HOD must first create departments, then manage personal and their personal information. This is critical for students who want to request an outpass of absence, for example.

The system needs the user to fill out several areas such as the user name and a brief form with a short explanation. In terms of history, the system shows all of the accessible outpass histories, together with the facts and status of each Student.

Finally aim of the project is to make a complete, fully working web based outpass system for students in hostel.

System to improve students everyday outpass management and to increase working efficiency.

References

- Online outpass management project in Django with outpass management
- <https://google.com>
- Outpass management project in C++ with source code