# INTRODUCTION

A college finder system main objective of that students can finding the their suitable college via this project, the student who are finishing the school they will their college based on marks, its very useful for the students. Admin login and added college details and which could be automatically search and give result to the students. Everyone can using this software, it will be more comfortable to use it.. The project is aimed to develop by JAVA as Front end and SQL SERVER as Back end. The back end is used to store the information in this system.

## 1.1. Organization Profile

ELife covers personal, business and corporate transportation needs, with a complete range of transportation options: from sedans to buses and luxury vehicles, in airport transportation services, local and long-distance transfers, hourly services, private tours and a wide etcetera.

Through a wide network of experienced drivers, ELife offers great advantages when booking ground transportation: multilingual attention, route and flight tracking, 24-hour call center, no hidden charges and more.

A fast-growing start-up headquartered in San Francisco, CA, USA in the heart of Silicon Valley. We recruit worldwide as our customer base is global. All employees outside San Francisco headquarter work remotely.

Elife is a Startup developing innovative technology in the passenger transport industry, providing any type of ground transportation worldwide, including Group/event transportation, NEMT transportation, senior and disability transportation, pet relocation, bilingual driver for non-English speak clients. Our market is the world. Our customers are everywhere. Our platform and apps are built on the cloud. We have endless exciting problems to solve and we only use the best and latest technology.

## System Specifications

### HARDWARE CONFIGURATION

**Processor** : Pentium -IV

**Speed** : 1 GHz

**Hard Disk Capacity** : 40GB

**RAM Capacity** : 1GB RAM

**CD-ROM Drive** : 52x speed

**Keyboard** : 104 keys

**Mouse** : Logitech

**Printer** : HP3745 series DeskJet printer

### SOFTWARE SPECIFICATION

**Operating System** : Windows 7/8/10

**Front End** : JAVA

**Back End** : SQL

**Feasibility Study**

# SYSTEM STUDY

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

The feasibility of a proposed solution is evaluated in teams of its components. These components are:

* + - * Economic feasibility
      * Technical feasibility

## Economic Feasibility

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

## Technical Feasibility

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

## EXISTING SYSTEM

The Existing system of college finder the student has to search the whole college in web site but it will not give proper details about the college also it could not shown the specific college based on the student 12 th mark. This system is very hard to find the college details. This is not possible to view the college details in same place. So the proposed system will help to avoid the drawbacks.

### DRAWBACKS

* Very difficult to search college based on their marks
* Difficult for the admin to manage the student and college details

## PROPOSED SYSTEM

The system automates the manual procedure of applying for admission in colleges . No need for students to go to different college for applying for admissions or enquiry. They can get a brief list of all the colleges at a single place. The system acts as a mediator between the student and college. It notifies the colleges of students enrolling for admission by emailing them so that it becomes easier for them to get students details and evaluate.

### FEATURES

* + - * Student can easy to identify their favorite college details
      * Admin has register the college details systematically.

# SYSTEM DESIGN AND DEVELOPMENT

Design is concerned with identifying software components specifying relationship Among components. Specifying software structure and providing blue print for the document phase. Modularity is one of the desirable properties of large systems. It implies that the system is divided into several parts. In such a manner, the interaction between parts is Minimal clearly specified. Design will explain software components in details. This will help the implementation of the system. Moreover, this will guide the further changes in the system to satisfy the further requirements.

The design document describes how to transform, the requirement and the functional design into more technical system design specification. This design involves conceiving and planning out in the mind and making a drawing pattern of sketch of. It includes type of activities, External Design, Architectural Design and Detailed Design. The architectural design and detailed design collectively referred to as internal design.

The external design involves specifying the externally observable characteristics of a software product and the internal design involves specifying the internal structure and processing details of the system. The fundamental concept of the design includes abstraction structure, information hiding Modularity, concurrency, verification and design aesthetics.

## FILE DESIGN

In computing, a file design (or file system) is used to control how data is stored and retrieved. Without a file system, information placed in a storage area would be one large body of data with no way to tell where one piece of information stops and the next begins. By separating the data into individual pieces, and giving each piece a name, the information is easily separated and identified. Taking its name from the way paper-based information systems are named, each group of data is called a "file". The structure and logic rules used to manage the groups of information and their names are called a "file system".

Some file systems are used on local data storage devices; others provide file access via a network protocol. Some file systems are "virtual", in that the "files" supplied are computed on request or are merely a mapping into a different file system used as a backing store. The file system manages access to both the content of files and the metadata about those files. It is responsible for arranging storage space; reliability, efficiency, and tuning with regard to the physical storage medium are important design considerations.

## INPUT DESIGN

The input design is the process of entering data to the system. The input design goal is to enter to the computer as accurate as possible. Here inputs are designed effectively so that errors made by the operations are minimized.

The inputs to the system have been designed in such a way that manual forms and the inputs are coordinated where the data elements are common to the source document and to the input. The input is acceptable and understandable by the users who are using it.

Input design is the process of converting user-originated inputs to a computer-based format input data are collected and organized into group of similar data. Once identified, appropriate input media are selected for processing.

The input design also determines the user to interact efficiently with the system. Input design is a part of overall system design that requires special attention because it is the common source for data processing error. The goal of designing input data is to make entry easy and free from errors.

Input design is the process of connecting the user-originated inputs into a computer to used format.

The goal of the input design is to make the data entry logical & free from errors.

## OUTPUT DESIGN

Output design is the process of converting computer data into hard copy that is understood by all. The various outputs have been designed in such a way that they represent the same format that the office and management used to.

Computer output is the most important and direct source of information to the user. Efficient, intelligible output design should improve the systems relationships with the user and help in decision making. A major form of output is the hardcopy from the printer.

Output requirements are designed during system analysis. A good starting point for the output design is the Data Flow Diagram (DFD). Human factors educe issues for design involves addressing internal controls to ensure readability.

The output form in the system is either by screen or by hard copies. Output design aims at communicating the results of the processing of the users. The reports are generated to suit the needs of the users. The reports have to be generated with appropriate levels.

All reports are output formats, maintained details can be reported over crystal reports, this project sustain following reports

## DATABASE DESIGN

The most important consideration in designing the database is how information will be used.

The main objectives of designing a database are:

### Data Integration

In a database, information from several files are coordinated, accessed and operated upon as through it is in a single file. Logically, the information are centralized, physically, the data may be located on different devices, connected through data communication facilities.

### Data Integrity

Data integrity means storing all data in one place only and how each application to access it. This approach results in more consistent information, one update being sufficient to achieve a new record status for all applications, which use it. This leads to less data redundancy; data items need not be duplicated; a reduction in the direct access storage requirement.

### Data Independence

Data independence is the insulation of application programs from changing aspects of physical data organization. This objective seeks to allow changes in the content and organization of physical data without reprogramming of applications and to allow modifications to application programs without reorganizing the physical data.

The tables needed for each module were designed and the specification of each and every column was given based on the records and details collected during record specification of the system study.

## SYSTEM DEVELOPMENT

The key to control maintenance costs is to design systems that are easy to change, so the link between development and maintenance is very strong. Many of the analysis and design methodologies, tools, and techniques employed during system development can be applied to system maintenance, but there are significant differences between development and maintenance. Maintainability is the ease with which software can be understood, corrected, adopted and enhanced.

### DESCRIPTION OF MODULES

To develop this project several step should be followed. There are various modules in this proposed system they are listed below.

* + - * Admin Login
      * Student Login
      * College Registration
      * Student Registration
      * Find College Details

### Admin Login:

The system will be under sole control of an admin who provides inputs to the system. Admin will update or add/delete different colleges from the system.

### Student Login:

Any student can login into the system by creating an account. He has to provide all the required information that is needed by a college by filling out a form.

### College Registration:

College details are only registered by admin they only have an access to create and enter the college details. When the admin has registered the college details it has to be display in the student home page.

### Student Registration:

Student can register their account self-way, before creating the account they should select the registration page and give their details. This screen will collect all the information about student and qualification details. once they enter the details it will store in the database.

### Find College Details:

Student can view the college details, first select the college and view that college details. Based on the student marks it will list out the college details as per the record. Also this was already stored in the college details database.

# TESTING AND IMPLEMENTATION

System testing is the process of exercising software with the intent of finding and ultimately correcting errors. This fundamental philosophy does not change for web applications, because Web-based systems and application reside on a network and interoperate with many different operating system, browsers, hardware platforms, and communication protocols; the search for errors represents a significant challenge for web application.

The distributed nature of client\server environments, the performance issues associated with transaction processing, the potential presence of a number of different hardware platforms, the complexities of network communication, the need to serve multiple clients from a centralized database and the requirements imposed on the server all combine to make testing of client\server architectures.

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer based system. System testing is the state of implementation that is aimed at assuring that the system works accurately and efficiently. Testing is the vital to the success of the system. System testing makes the logical assumption that if all the parts of the system are correct, the goal will be successfully achieved.

### Unit Testing

Unit testing focuses verification efforts on the smallest unit of software design of the module. This is also known as “module testing”. This testing is carried out during programming stage itself. In this testing step, each module is found to be working satisfactorily as regards to the expected output of the modules.

### Integration Testing

Data can be lost across an interface, one module can have adverse effect on another sub function when combined it may not produce the desired major functions. Integration testing is a systematic testing for constructing test to uncover errors associated within an interface.

The objectives taken from unit tested modules and a program structure is built for integrated testing.

All the modules are combined and the test is made.

A correction made in this testing is difficult because the vast expenses of the entire program complicated the isolation of causes. In this integration testing step, all the errors are corrected for next testing process.

### Validation Testing

After the completion of the integrated testing, software is completely assembled as a package; interfacing error has been uncovered and corrected and a final series of software test validation begins.

Validation testing can be defined in many ways but a simple definition is that validation succeeds when the software function in a manner that can be reasonably expected by the customer. After validation test has been conducted, one of two possible conditions exists:

### Output Testing

The next process of validation testing, is output testing of the proposed system, since no system could be successful if it does not produce the required output in the specified format. Asking the user about the format required, list the output to be generated or displayed by the system under considerations.

Output testing is a different test whose primary purpose is to fully exercise the computer based system although each test has a different purpose all the work should verify that all system elements have been properly integrated and perform allocated functions.

The output format on the screen is found to be corrected as the format was designed in the system design phase according to the user needs for the hard copy also; the output testing has not resulted in any correction in the system.

**IMPLEMENTATION**

System implementation is the stage of the project that the theoretical design is turned into a working system. If the implementation stage is not properly planned and controlled, it can cause error. Thus it can be considered to be the most crucial stage in achieving a successful new system and in giving the user confidence that the new system will work and be effective.

Normally this stage involves setting up a coordinating committee, which will act as a sounding board for ideas; complaints and problem. The first task is implementation planning; i.e., deciding on the methods and time scale to be adopted. Apart from planning two major task of preparing for implementation are, education takes place much earlier in the project; at the implementation stage the emphasis must be on training in new skills to give staff confidence they can use the system. Once staff has been trained, the system can be tested.

After the implementation phase is completed and the user staff is adjusted to the changes created by the candidate system, evaluation and maintenance is to bring the new system to standards.

# CONCLUSION

In conclusion, a college finder system is a powerful tool for students and their families to navigate the college search process. The system helps students find colleges that match their academic goals, personal preferences, and financial resources, and provides tools to organize and track the application process.

By using a college finder system, students can save time and reduce stress by having all of the necessary information and tools in one place. The system can help students stay on track with application deadlines, track their progress, and receive reminders about important tasks.

In addition, the college finder system can provide valuable information about colleges, including admission requirements, student demographics, and financial aid options. This information can help students make informed decisions about where to apply and which colleges are the best fit for their needs.

Overall, a college finder system is an essential tool for anyone navigating the college search process. It can simplify the process, save time, and provide valuable information that can help students make informed decisions about their future..

## BIBLIOGRAPY

**Books Referred :**

* Lea, Doug. Concurrent Programming in Java: Design Principles and Patterns. Addison-Wesley, 2000.
* Naftalin, Maurice, and Philip Wadler. Java Generics and Collections. O'Reilly Media, 2006.
* Shildt, Herbert. Java: A Beginner's Guide. McGraw-Hill Education, 2018.
* Sierra, Kathy, and Bert Bates. Head First Java, 2nd Edition. O'Reilly Media, 2005.
* Subramaniam, Venkat. Functional Programming in Java: Harnessing the Power of Java 8 Lambda Expressions. Pragmatic Bookshelf, 2014.
* Bloch, Joshua. Effective Java: Programming Language Guide. Addison-Wesley, 2017.

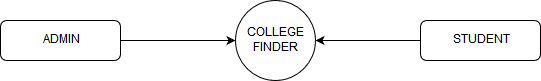
## Websites:

* JavaWorld. "Java Tutorials and Articles." JavaWorld, https://www.javaworld.com/.
* JetBrains. "IntelliJ IDEA Documentation." JetBrains, <https://www.jetbrains.com/help/idea/>.
* Spring Framework. "Spring Boot Documentation." Spring Framework, <https://docs.spring.io/spring-boot/docs/current/reference/html/>.
* Baeldung. "Java Tutorials and Articles." Baeldung, 2023, https://www.baeldung.com/java-tutorials.

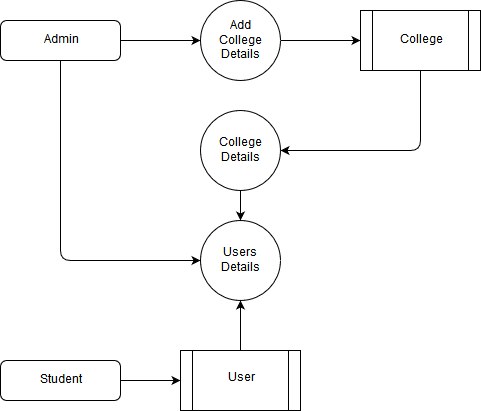
# APPENDICES

## Data Flow Diagram

**Level 0**



**Level 1**

****

## TABLE STRUCTURE

**Table Name :** Admin

**Primary Key :** Admin\_id

**Table Description :** This table is used to maintain the details about admin

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| Admin\_id | Int | 8 | Primary Key |
| Username | Varchar | 25 | Not null |
| Address | Varchar | 50 | Not null |
| Phone\_no | Varchar | 15 | Not null |
| Mobile\_No | Varchar | 15 | Not null |

**Table Name :** College

**Primary Key :** College\_id

**Table Description :** This table is used to maintain the details about college

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| college\_id | Int | 8 | Primary Key |
| Institution | Varchar | 25 | Not null |
| Name | Varchar | 10 | Not null |
| Min mark | Int | 10 | Not null |
| Address | Varchar | 50 | Not null |
| Link | Varchar | 30 | Not null |
| City | Varchar | 10 | Not null |
| District | Varchar | 10 | Not null |
| Pincode | Int | 10 | Not null |

**Table Name :** Student

**Primary Key :** Student\_id

**Table Description :** This table is used to maintain the details about student

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **SIZE** | **CONSTRAINT** |
| Student\_id | Int | 8 | Primary Key |
| Email | Varchar | 25 | Not null |
| Password | Varchar | 10 | Not null |
| Name | Varchar | 10 | Not null |
| School | Varchar | 25 | Not null |
| Mark | Int | 10 | Not null |
| Inst | Varchar | 30 | Not null |
| Community | Varchar | 10 | Not null |

## B. Sample Coding

var express = require('express');

var router = express.Router();

var knex = require('knex');

var knex = require('knex')({

client: 'mysql',

connection: {

host: 'localhost',

user: 'root',

password: '',

database: 'college\_finder'

}

});

router.post('/login', function (req, res, next) {

var query = "select count(\*) as count from admin where username = '" + req.body.username + "' and password='" + req.body.password + "'";

knex.raw(query).then((datas)=> {

var count = datas[0][0].count;

if (count == 0) {

var query = "select count(\*) as count , id as id from student where email = '" + req.body.username + "' and password='" + req.body.password + "'";

console.log(query)

knex.raw(query).then((datas)=> {

var cou = datas[0][0].count;

if (cou != 0) {

var id = datas[0][0].id;

res.send(id + "")

} else {

res.send('Invalid username and password')

}

});

} else {

res.send('admin')

}

})

});

router.get('/get\_institutions', function (req, res, next) {

var query = "select \* from institutions ";

knex.raw(query).then((datas)=> {

res.send(datas[0])

})

});

router.post('/add\_college', function (req, res, next) {

var name = req.body.name;

var address = req.body.address;

var bc = req.body.bc;

var mbc = req.body.mbc;

var oc = req.body.oc;

var sc = req.body.sc;

var link\_address = req.body.link\_address;

var inst = req.body.inst;

var city = req.body.city;

var district = req.body.district;

var pincode = req.body.pincode;

var query = "INSERT INTO `college` (`id`, `inst`, `name`, `address`, `bc`, `mbc`, `oc`, `sc`, `linkaddress`,`city`,`district`,`pincode`) VALUES (NULL, '" + inst + "', '" + name + "', '" + address + "', '" + bc + "', '" + mbc + "', '" + oc + "', '" + sc + "', '" + link\_address + "', '" + city + "', '" + district + "', '" + pincode + "');";

knex.raw(query).then((datas)=> {

res.send("College has been added sucessfully")

})

});

router.get('/get\_district', function (req, res, next) {

var query = 'SELECT district from college GROUP by district';

knex.raw(query).then((datas)=> {

res.send(datas[0])

})

});

router.get('/get\_college/:district', function (req, res, next) {

var query = 'SELECT \* from college where district="' + req.params.district + '"';

knex.raw(query).then((datas)=> {

res.send(datas[0])

})

});

router.get('/get\_college\_details/:id', function (req, res, next) {

var query = 'SELECT \* from college where id="' + req.params.id + '"';

knex.raw(query).then((datas)=> {

res.send(datas[0])

})

});

router.get('/get\_student\_profile/:id', function (req, res, next) {

var query = 'SELECT \* from student where id="' + req.params.id + '"';

knex.raw(query).then((datas)=> {

res.send(datas[0])

})

});

router.get('/get\_colleges\_marks/:mark/:community', function (req, res, next) {

var community = req.params.community;

var query = 'SELECT \* from college where '+community+' <=' + req.params.mark ;

console.log(query)

knex.raw(query).then((datas)=> {

res.send(datas[0])

})

});

router.get('/getStudentList', function (req, res, next) {

var query = 'SELECT \* from student';

knex.raw(query).then((datas)=> {

res.send(datas[0])

})

});

router.post('/add\_customer', function (req, res, next) {

var email = req.body.email;

var password = req.body.password;

var name = req.body.name;

var school = req.body.school;

var mark = req.body.mark;

var community = req.body.community;

var inst = '1';

var query = "INSERT INTO `student` (`id`, `email`, `password`, `name`, `school`, `mark`, `inst`,`community`) VALUES (NULL, '"+email+"', '"+password+"', '"+name+"', '"+school+"', '"+mark+"', '"+inst+"', '"+community+"');";

knex.raw(query).then((datas)=> {

res.send("Student has been created")

})

});

module.exports = router;

// Material Design example

var selected\_institution;

$(document).ready(function () {

update();

});

function update() {

getInstitutions();

getDistrict();

getStudentProfile();

getStudentList();

}

function getInstitutions() {

$.ajax({

type: "GET",

url: "/api/get\_institutions",

success: function (result) {

var html = ``;

for (var i in result) {

var data = result[i];

html += ` <option id="` + data.id + `">` + data.name + `</option>

`;

}

$("#inst\_1").html(html)

}

});

}

function addCollege() {

var name = $("#name1").val();

var address = $("#address1").val();

var bc = $("#bc").val();

var mbc = $("#mbc").val();

var oc = $("#oc").val();

var sc = $("#sc").val();

var link\_address = $("#link\_address1").val();

var city = $("#city1").val();

var district = $("#district1").val();

var pincode = $("#pincode1").val();

var inst = $("#inst\_1").children(':selected').attr('id');

$.ajax({

type: "POST",

url: "/api/add\_college",

data: {

name: name,

address: address,

bc: bc,

mbc: mbc,

oc: oc,

sc: sc,

link\_address: link\_address,

inst: inst,

city: city,

district: district,

pincode: pincode,

},

success: function (result) {

alert(result);

$("#name1").val("");

$("#address1").val("");

$("#min\_mark1").val("");

$("#link\_address1").val("");

$("#city1").val("");

$("#district1").val("");

$("#pincode1").val("");

update();

}

});

}

function getDistrict() {

$.ajax({

type: "GET",

url: "/api/get\_district",

success: function (result) {

var html = ``;

for (var i in result) {

var data = result[i];

html += ` <option>` + data.district + `</option>

`;

}

$("#district2").html(html)

getCollege();

}

});

}

function getCollege() {

var district = $("#district2").val();

$.ajax({

type: "GET",

url: "/api/get\_college/" + district,

success: function (result) {

var html = ``;

for (var i in result) {

var data = result[i];

html += ` <option id="` + data.id + `">` + data.name + `</option>

`;

}

$("#college2").html(html)

}

});

}

function getCollegeDetails() {

var college\_id = $("#college2").children(':selected').attr('id');

$.ajax({

type: "GET",

url: "/api/get\_college\_details/" + college\_id,

success: function (result) {

var html = ``;

for (var i in result) {

var data = result[i];

html += `<div class="card-body">

<B><h5 class="card-title">` + data.name + `</h5>

<p class="card-text">` + data.address + `</p><br>

<p>` + data.city + `</p><br>

<p>` + data.district + `</p>

<p>` + data.pincode + `</p>

<a href=` + data.linkaddress + `>` + data.linkaddress + `</a>

</B>

</div>`;

}

$("#college\_details\_body").html(html)

}

});

}

function getStudentProfile() {

var id = getUrlParameter('id');

$.ajax({

type: "GET",

url: "/api/get\_student\_profile/" + id,

success: function (result) {

$("#tot2").val(result[0].mark)

$("#community2").val(result[0].community)

}

});

}

function searchColleges() {

var mark = $("#tot2").val();

var community = $("#community2").val();

$.ajax({

type: "GET",

url: "/api/get\_colleges\_marks/" + mark+"/"+community,

success: function (result) {

var html = '';

for (var i in result) {

var data = result[i];

html += ` <div class="card" style="border: double;margin: 10px" >

<div style="padding: 10px" class="card-body">

<B><h5 class="card-title">` + data.name + `</h5>

<p class="card-text">` + data.address + `</p><br>

<p>` + data.city + `</p><br>

<p>` + data.district + `</p>

<p>` + data.pincode + `</p>

<a href=` + data.linkaddress + `>` + data.linkaddress + `</a>

</B>

</div> </div>`;

}

$("#mycollege\_body").html(html)

}

});

}

var getUrlParameter = function getUrlParameter(sParam) {

var sPageURL = window.location.search.substring(1),

sURLVariables = sPageURL.split('&'),

sParameterName,

i;

for (i = 0; i < sURLVariables.length; i++) {

sParameterName = sURLVariables[i].split('=');

if (sParameterName[0] === sParam) {

return sParameterName[1] === undefined ? true : decodeURIComponent(sParameterName[1]);

}

}

};

function getStudentList() {

$.ajax({

type: "GET",

url: "/api/getStudentList" ,

success: function (result) {

var html = '';

for (var i in result) {

var data = result[i];

html += ` <tr>

<th scope="row">`+(++i)+`</th>

<td>`+data.name+`</td>

<td>`+data.email+`</td>

<td>`+data.school+`</td>

<td>`+data.mark+`</td>

</tr>`;;

}

$("#student\_details\_body").html(html)

}

});

}

var createError = require('http-errors');

var express = require('express');

var path = require('path');

var cookieParser = require('cookie-parser');

var logger = require('morgan');

var engines = require('consolidate');

var indexRouter = require('./routes/index');

var usersRouter = require('./routes/api');

var app = express();

// view engine setup

app.set('views', path.join(\_\_dirname, 'views'));

app.engine('html', engines.mustache);

app.set('view engine', 'html');

app.use(logger('dev'));

app.use(express.json());

app.use(express.urlencoded({ extended: false }));

app.use(cookieParser());

app.use(express.static(path.join(\_\_dirname, 'public')));

app.use('/', indexRouter);

app.use('/api', usersRouter);

// catch 404 and forward to error handler

app.use(function(req, res, next) {

next(createError(404));

});

// error handler

app.use(function(err, req, res, next) {

// set locals, only providing error in development

res.locals.message = err.message;

res.locals.error = req.app.get('env') === 'development' ? err : {};

// render the error page

res.status(err.status || 500);

res.render('error');

});

module.exports = app;

<html>

<head>

<link href="/bootstrap/bootstrap.css" rel="stylesheet"/>

<link href="/bootstrap/datatables.css" rel="stylesheet"/>

<link rel="stylesheet" type="text/css" href="/home/home.css">

</head>

<body>

<!-- Remove dark nav -->

<nav class="nav dark-nav">

<div class="container">

<div class="nav-heading">

<button class="toggle-nav" data-toggle="open-navbar1"><i class="fa fa-align-right"></i></button>

<a class="brand" href="#">College Finder</a>

</div>

<div class="menu" id="open-navbar1">

<ul class="list">

<li><a href="#htab1" data-toggle="tab">Add College</a></li>

<li><a href="#htab2" data-toggle="tab">College Details</a></li>

<li><a href="#htab3" data-toggle="tab">Student Details</a></li>

<li><a href="/">Logout</a></li>

</ul>

</div>

</div>

</nav>

var express = require('express');

var router = express.Router();

var knex = require('knex');

var knex = require('knex')({

client: 'mysql',

connection: {

host: 'localhost',

user: 'root',

password: '',

database: 'college\_finder'

}

});

router.post('/login', function (req, res, next) {

var query = "select count(\*) as count from admin where username = '" + req.body.username + "' and password='" + req.body.password + "'";

knex.raw(query).then((datas)=> {

var count = datas[0][0].count;

if (count == 0) {

var query = "select count(\*) as count , id as id from student where email = '" + req.body.username + "' and password='" + req.body.password + "'";

console.log(query)

knex.raw(query).then((datas)=> {

var cou = datas[0][0].count;

if (cou != 0) {

var id = datas[0][0].id;

res.send(id + "")

} else {

res.send('Invalid username and password')

}

});

} else {

res.send('admin')

}

})

});

router.get('/get\_institutions', function (req, res, next) {

var query = "select \* from institutions ";

knex.raw(query).then((datas)=> {

res.send(datas[0])

})

});

router.post('/add\_college', function (req, res, next) {

## D. Sample Input

## User Login page

## 

## Student login page & Registration Page

## 

## Input of student Registration

## 

## Input of login page

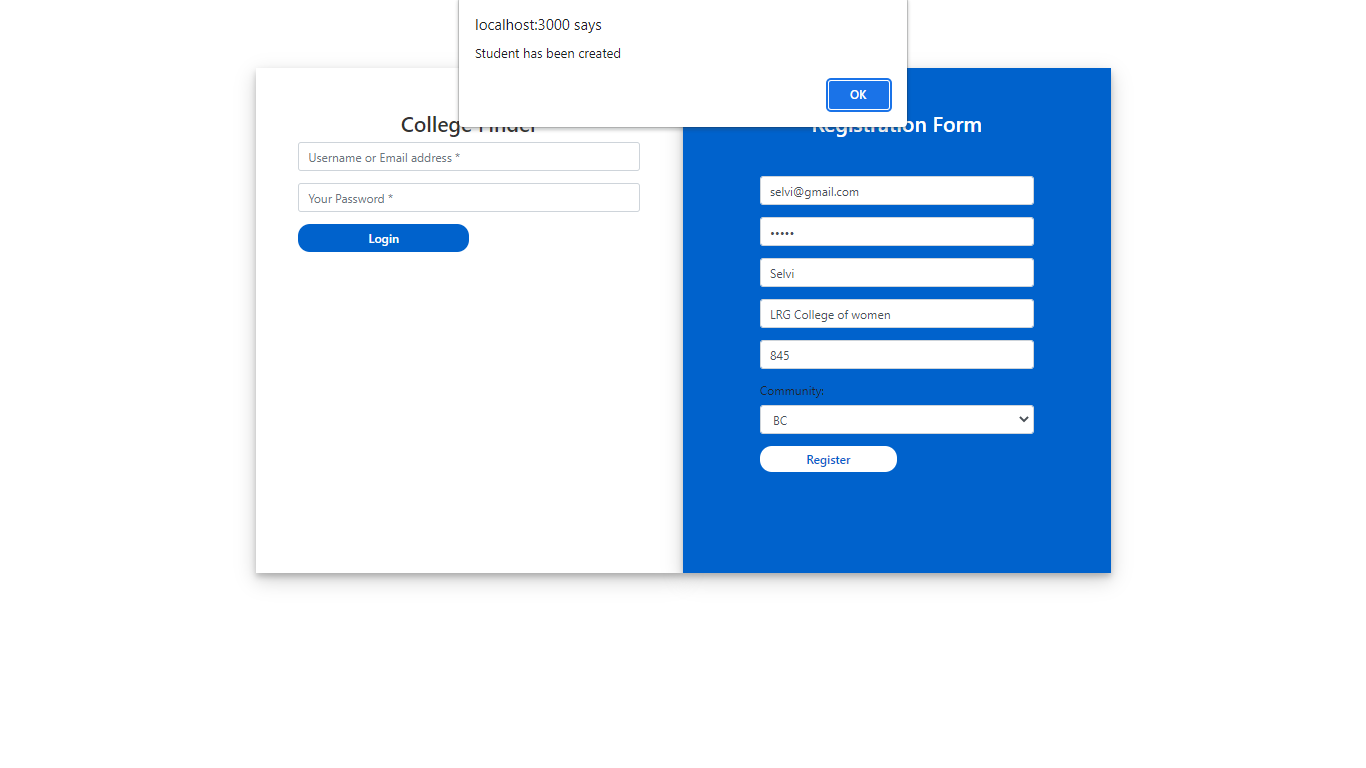
## 

## Input of college registration page for Admin

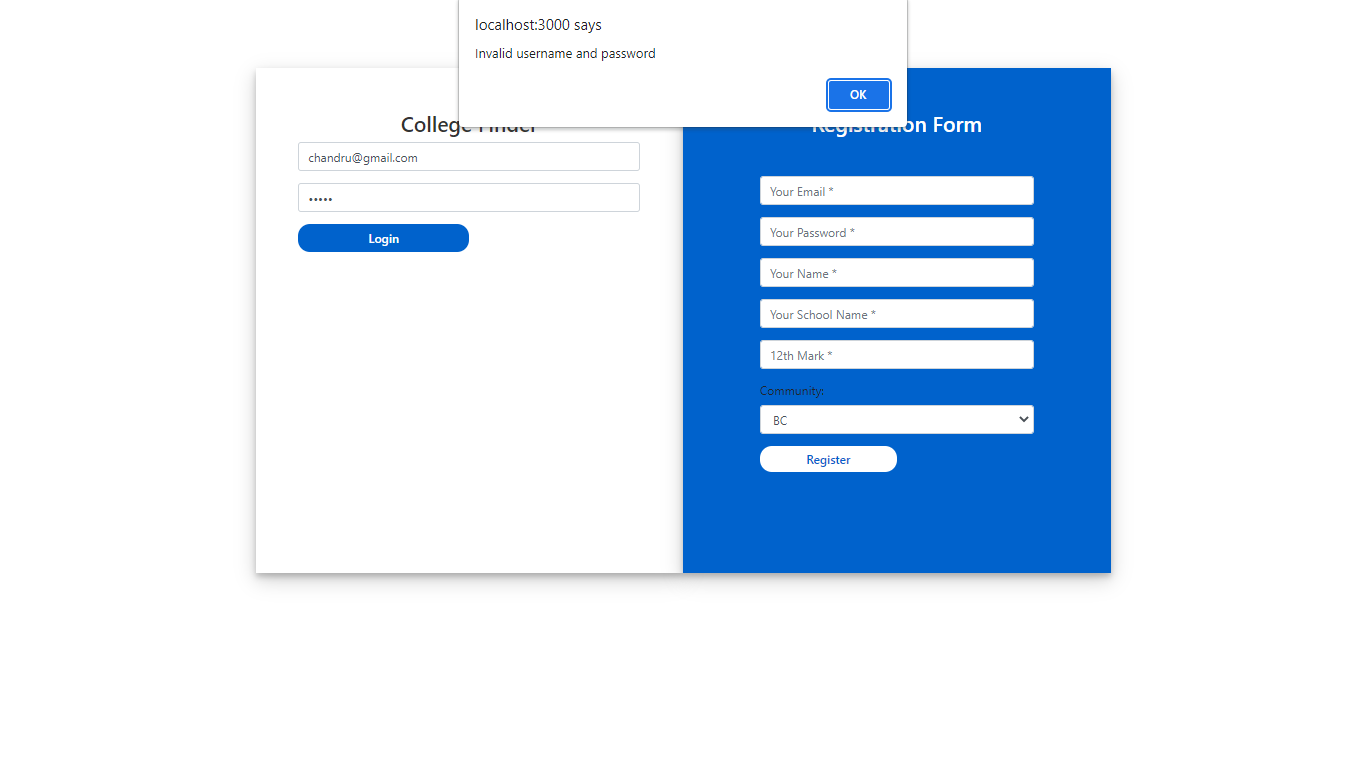
## 

## E. Sample Output

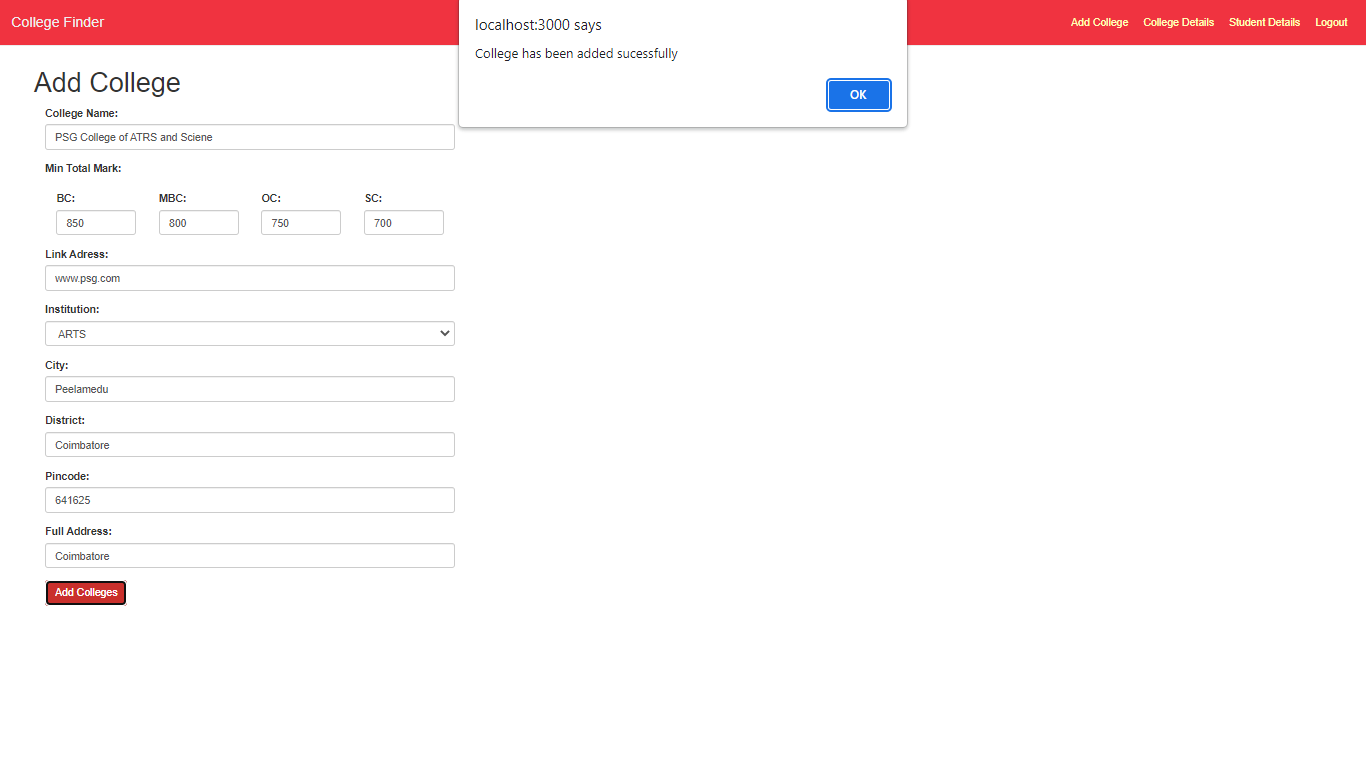
**Output of Student Registration**



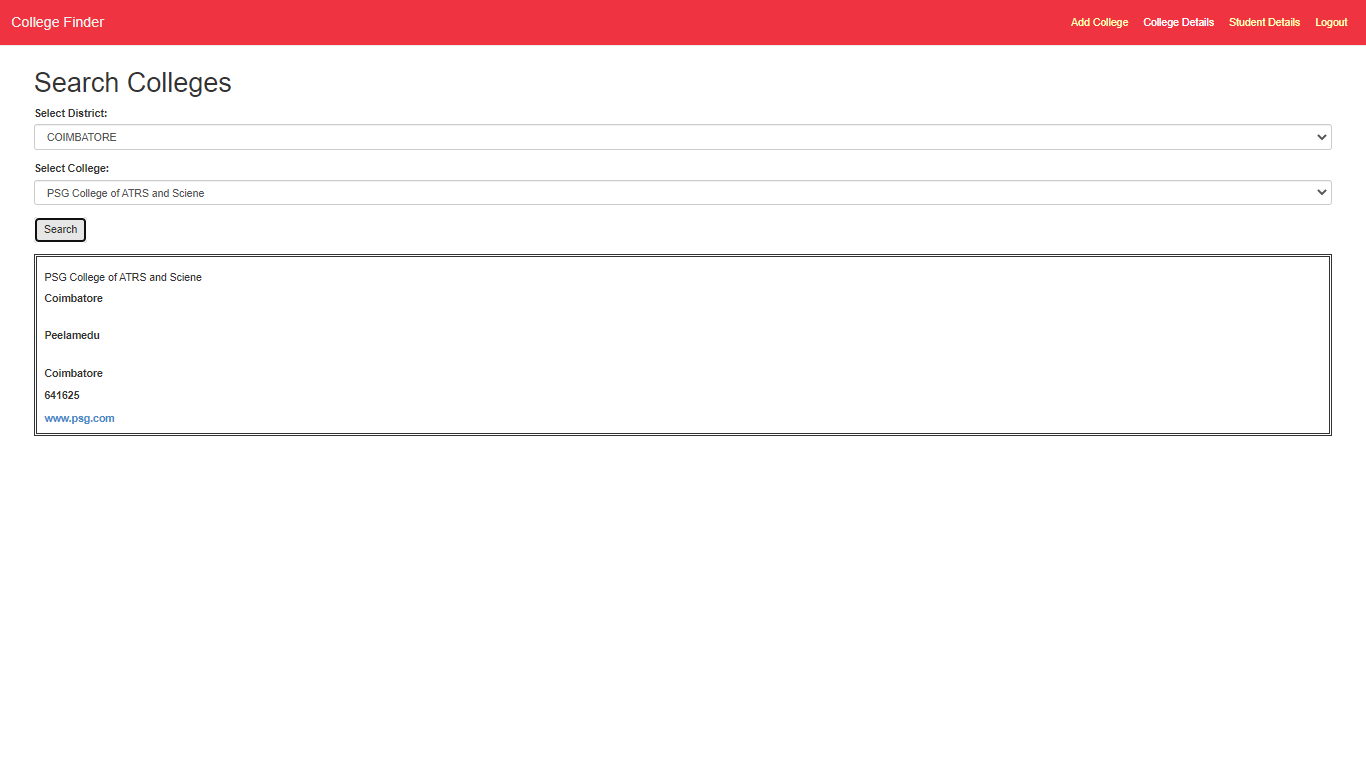
Output of Invalid username or password



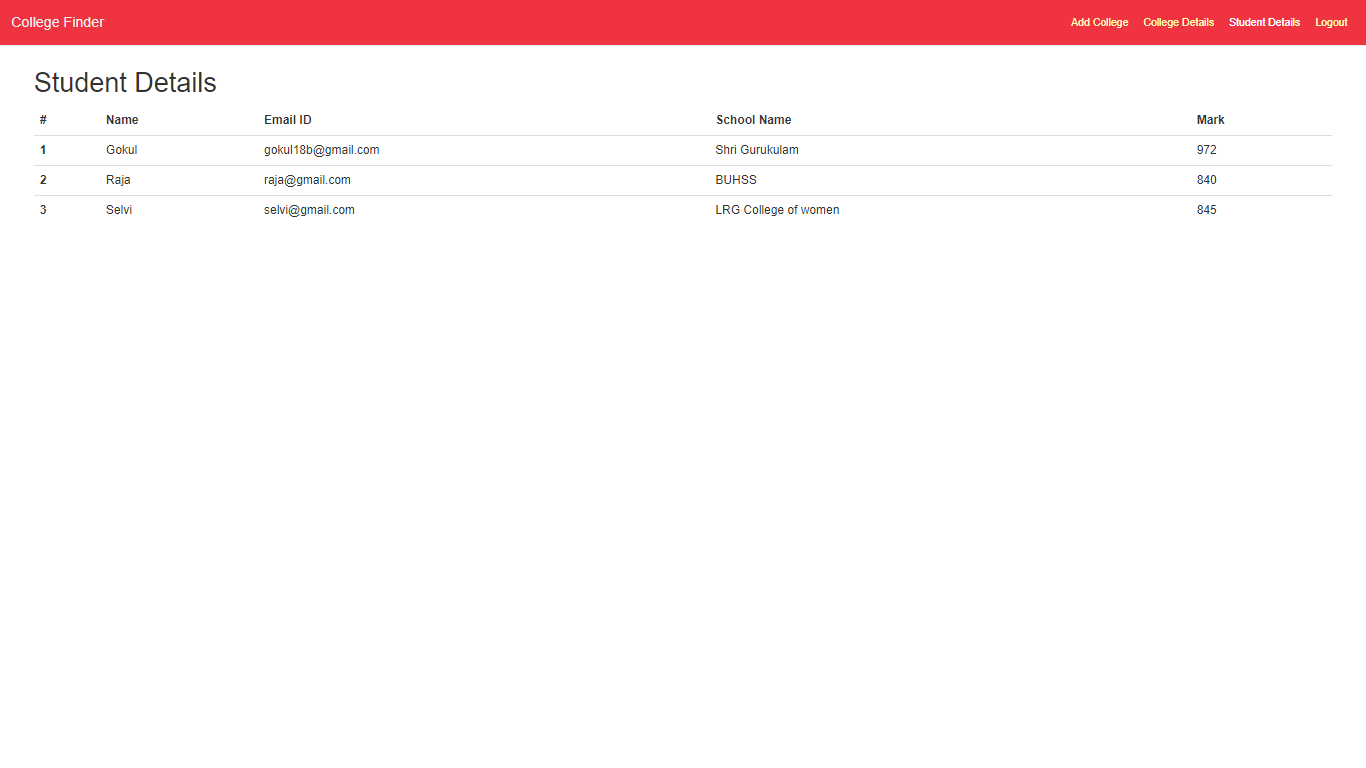
Output of college registration



Output of College details page



Output of Student details page



Output of Matched college details

