INTRA-INSTITUTE INTERNSHIP MANAGEMENT SYSTEM

Roll no.: 31330 - Sidhant Hargunani

Roll no.: 31332 - Aditya Jadhav

Roll no.: 31344 - Krisha Bhambani

Abstract

For a student, getting an external internship approved by the college is an unnecessarily lengthy and stressful process. It involves taking appointments with multiple teachers to show them the offer letter, your academic reports, and the reason you are the perfect fit for the internship. We propose a centralized, integrated system to completely automate this process. Our system allows students to enter their academic details (in the future we plan to fetch these details from college records), request approval for external internships as well as directly apply to college internships from the portal. Teachers can then approve or reject these applications based on the duration, feasibility, and academic record of the student. Apart from that, we have added functionality for teachers to add college-level internships to the portal as well. Students can apply to these with the click of a button.

Introduction

Our system provides multiple flows for both users of the system:

1. Student request approval of external internship

2. Student applying to college internship

3. Teacher add college internships

4. Teacher approving external internship applications

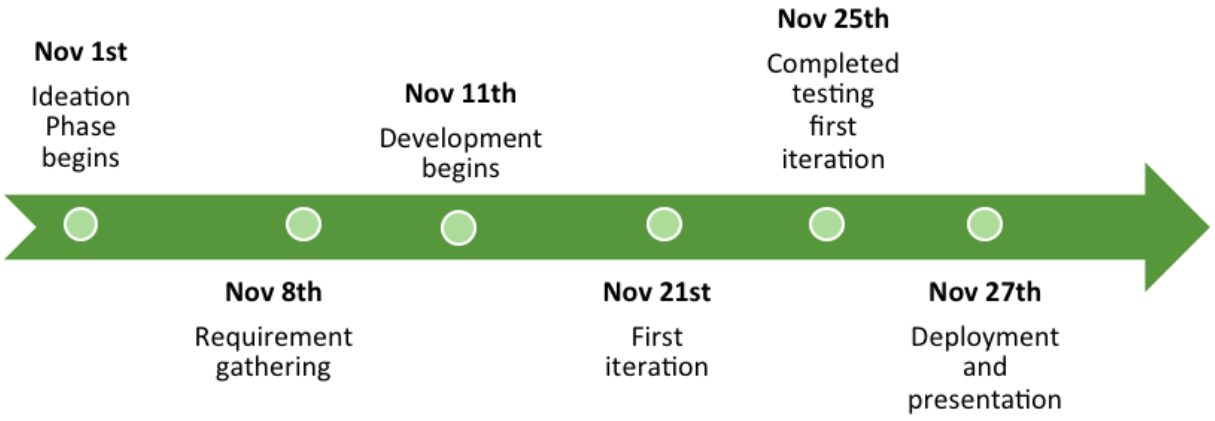
These flows make the internship formalities for students as well as teachers seamless.

Scope

Objectives of the project:

- Complete automation of the lengthy current internship approval process in colleges.
- Providing a centralized portal for teachers and the TnP cell to post internships, as well as for students to apply to them.
- Obtaining useful insights and reports of the number of students that have completed internships, stipends, as well as a list of companies that students chose to intern with.

Timeline:



USE CASE DIAGRAM

Functionalities:

User 1: Students

- Account login.
- Update student details like gpa, attendance, etc.
- Apply for student offered internships.
- Request approval for received internship.

User 2: Department level heads

- Approve/Deny internships for students based on accessible profiles, information.
- Post internship opportunities.
- Generate monthly, quarterly, and yearly record

Requirements

Tech stack

Frontend	Backend
• HTML • CSS • AngularJS	• Flask Database • SQLite3

Atomic requirements

User	Requirements
Student	• Login & register. • Add academic details. • View available internships (college – level). • Request approval from college for external internships. • Apply to college – level internships. • View status of college internship application. • View status of external internship approval applications.
Teacher / TnP Officer	• Login & register. • Add college – level internship. • View external internship applications. • View responses to college – level internships. • Generate reports - list of students that completed internships, details of internships completed.

Data Modelling & Relational Database Design

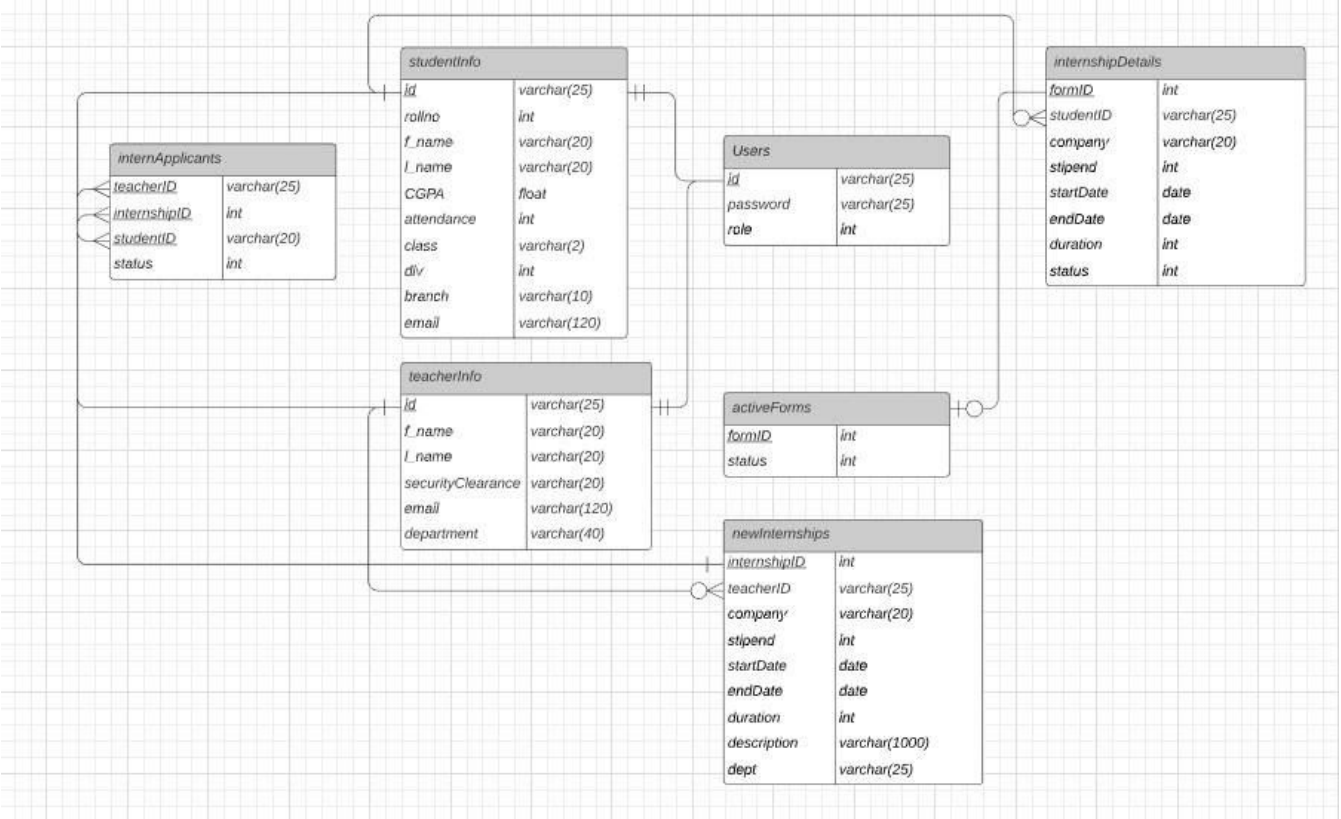


Figure 1 Entity Relationship Diagram for Internship Approval System

Data Dictionary

Field Name	Data Type	Data Format	Field Size	Description	Example
Id	Text		25	User id, primary key	C2K18109912
Password	Text		25	Login password of user	samplePass12*
Role	Integer		1	Role (Student / Teacher)	0

Data Dictionary for Users table. It stores the login credentials of all the users of the system.

Field Name	Data Type	Data Format	Field Size	Description	Example
Id	Text		25	Registration id, primary key	C2K18109912
Roll Number	Integer	NNNNN	5	Student roll number	31394
First Name	Text		20	Student first name	Walter
Last Name	Text		20	Student last name	White
CGPA	Float			Overall CGPA of student	9.84
Attendance	Float	NN.NN%		Annual attendance	91.45%
Class	Text	LL	2	Year	TE
Division	Integer			Student's division	03
Branch	Text		10	Course branch	Comp
Email	Text		120	Student's email	ww@gmail.com

Data Dictionary for Student Info table. This table stores the academic as well as personal details of the student

Field Name	Data Type	Data Format	Field Size	Description	Example
Id	Text		25	Teacher registration id, primary key	T2K28127915
First Name	Text		20	Teacher first name	Hank
Last Name	Text		20	Teacher last name	Schrader
Security Clearance	Text		20	Position of teacher in the institute	Department Head
Department	Text		10	Teaching department	Comp
Email	Text		120	Teacher's email	hs@gmail.com

Data Dictionary for Teacher Info table. This table stores details of a teacher that are necessary to allow posting internships as well as approving applications.

Field Name	Data Type	Data Format	Field Size	Description	Example
Internship ID	Integer			Internship id, primary key	1294121
Student ID	Text		25	Student registration number (FK)	C2K18109912
Company	Text		20	Company name	Google
Stipend	Integer		20	Monthly stipend	55000
Start Date	Date	YYYY-MM-DD		Internship start date	2020-09-08
End Date	Date	YYYY-MM-DD		Internship end date	2020-11-08
Duration	Integer			Duration in weeks	8
Status	Integer			Status code of application	2

Data Dictionary for Internship Details table. This table stores details of external internships that a student is requesting approval to undertake

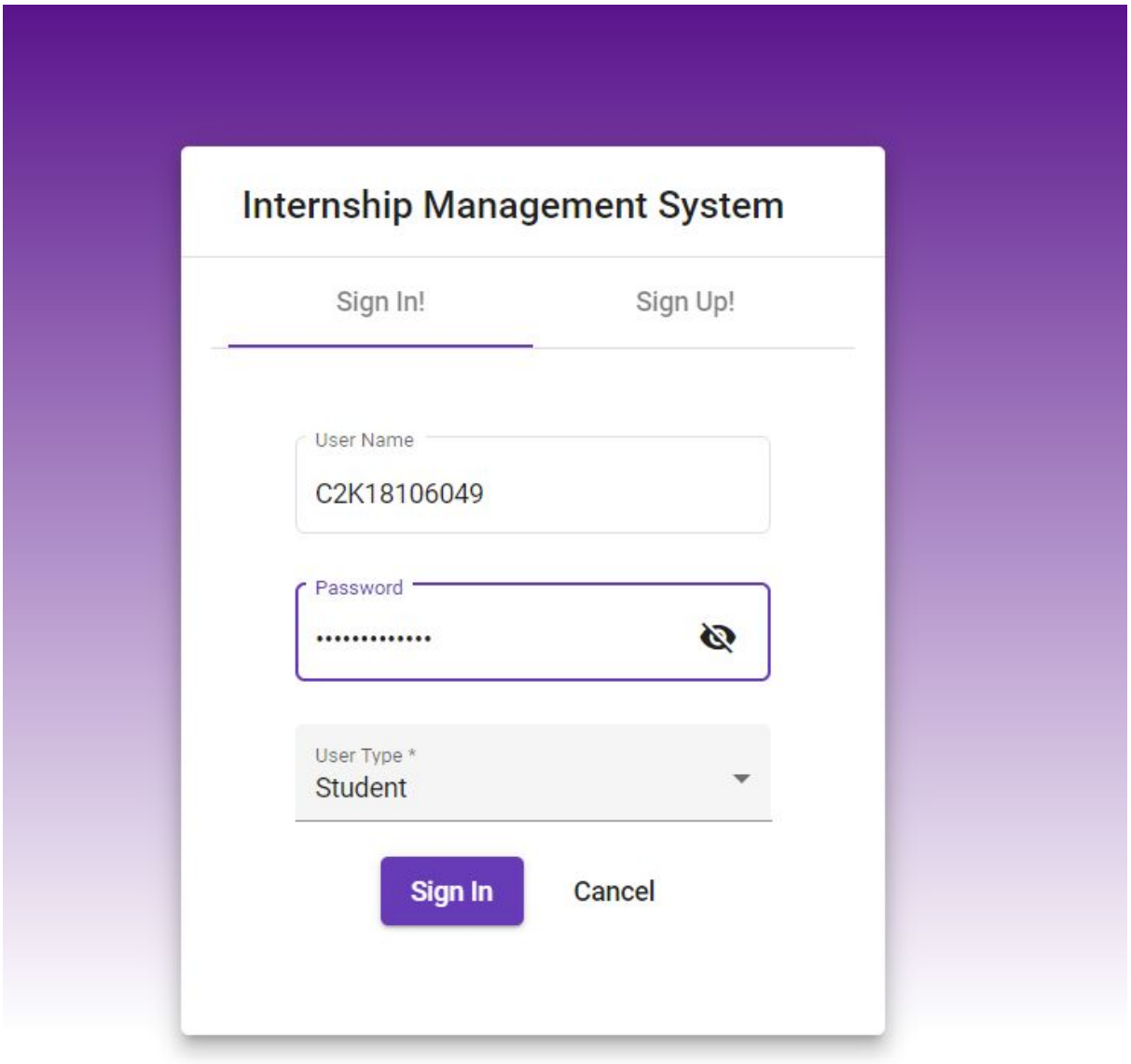
Field Name	Data Type	Data Format	Field Size	Description	Example
Teacher ID	Text		25	Teacher id, primary key (FK)	T2K28127915
Internship ID	Integer			Internship id, primary key (FK)	1294121
Student ID	Text		25	Student id, primary key (FK)	C2K18109912
Status	Integer		1	Status of application mapped to text	2

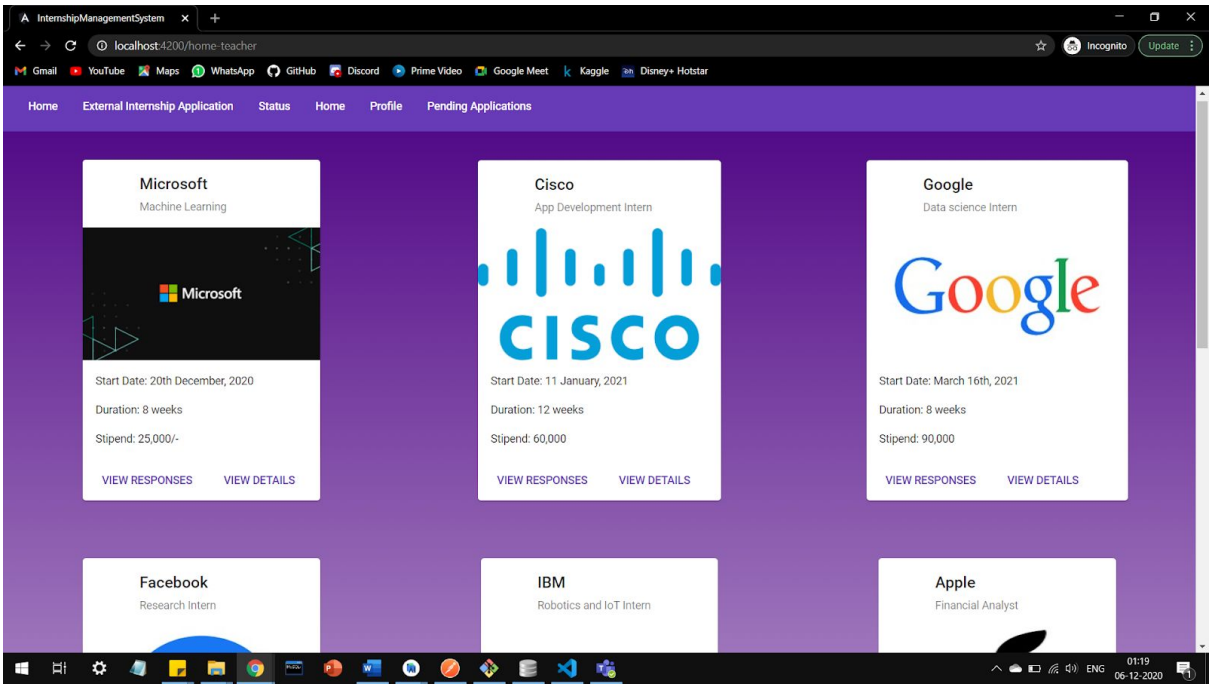
Data Dictionary for Intern Applicants table. This table maps student IDs to the applied internship, as well as the teacher who added the internship.

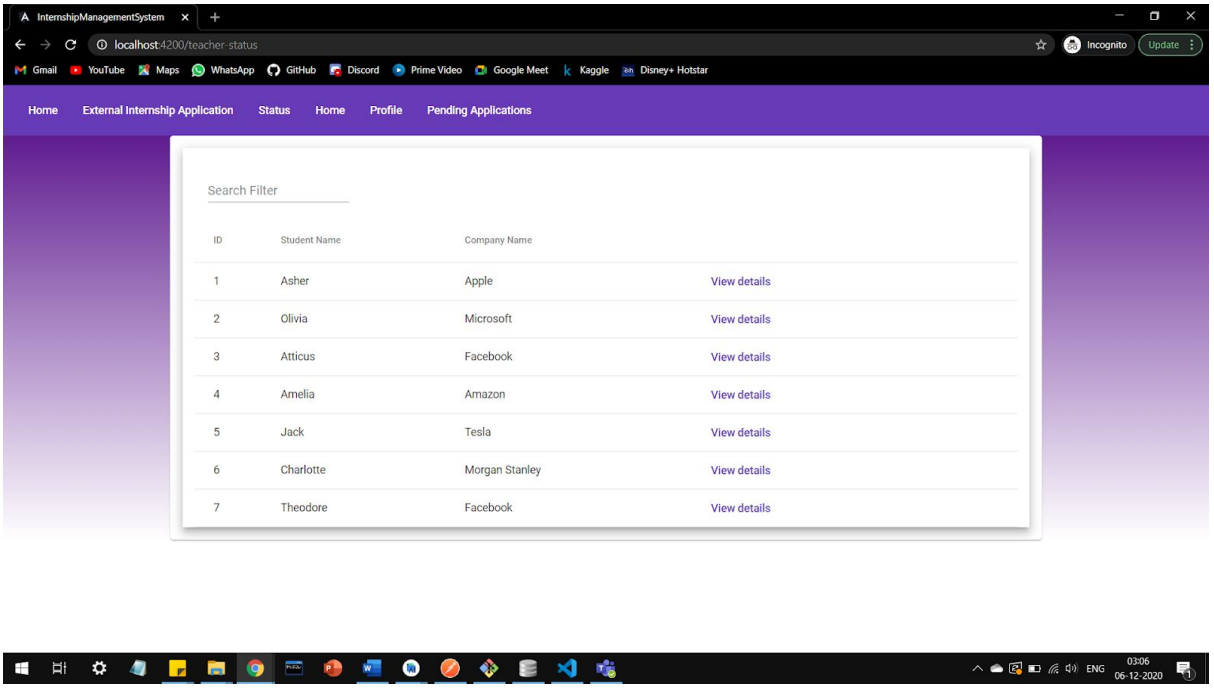
Field Name	Data Type	Data Format	Field Size	Description	Example
Internship ID	Integer			Internship id, primary key	1294121
Teacher ID	Text		25	Teacher ID (FK)	T2K28127915
Company	Text		20	Company name	Google
Stipend	Integer		20	Monthly stipend	55000
Start Date	Date	YYYY-MM-DD		Internship start date	2020-09-08
End Date	Date	YYYY-MM-DD		Internship end date	2020-11-08
Duration	Integer			Duration in weeks	8
Description	Text		1000	Detailed description of the internship	Data science intern at Google!

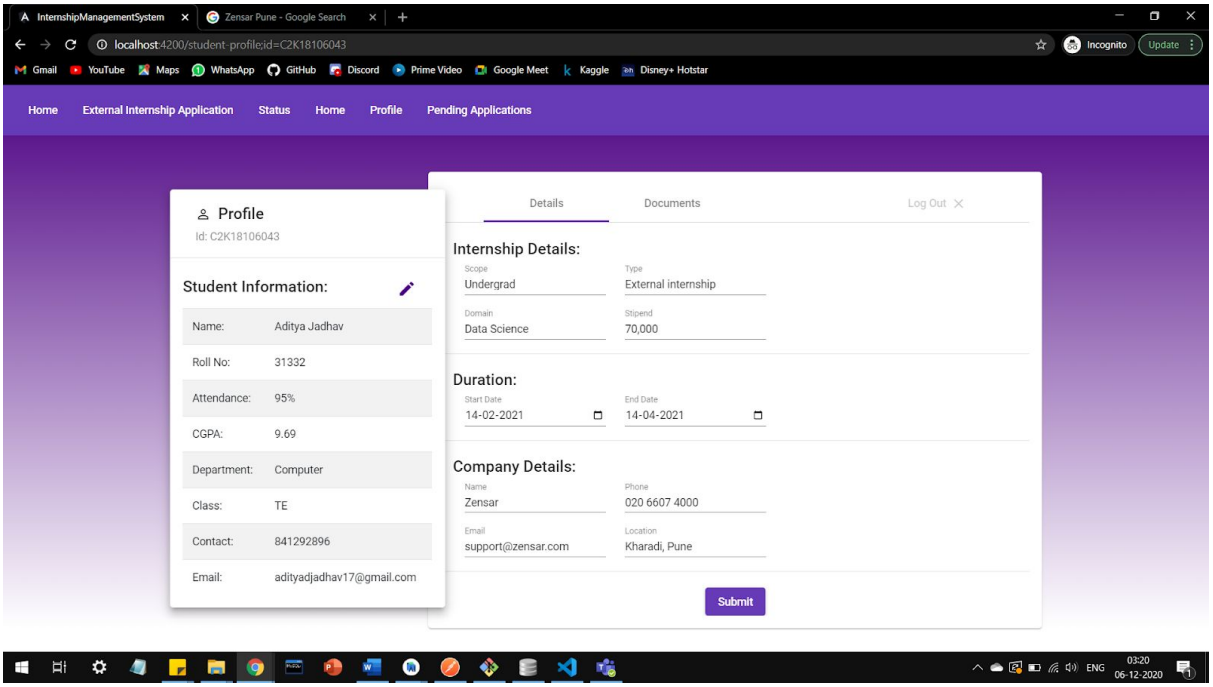
Data Dictionary for New Internships table. This table stores details of college-level internships added by teachers or the TnP officer. Students can apply to these internship

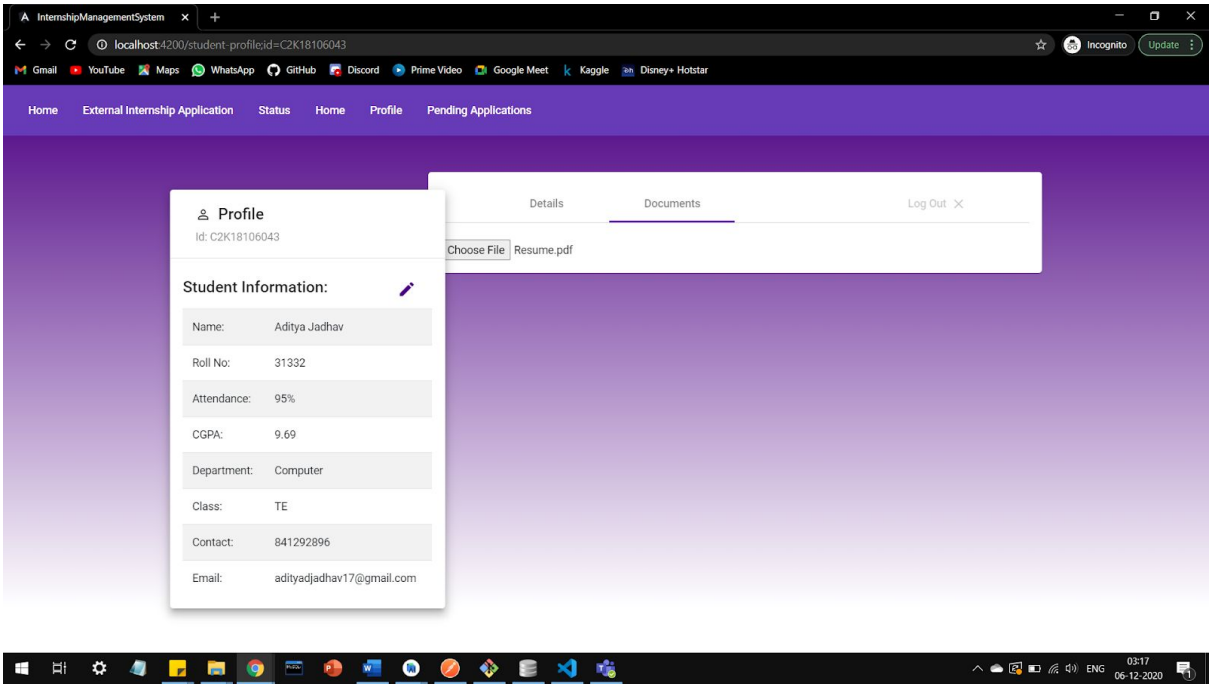
GRAPHICAL USER INTERFACE




InternshipManagementSystem
localhost:4200/home-teacher
Incognito
Update
Gmail
YouTube
Maps
WhatsApp
GitHub
Discord
Prime Video
Google Meet
Kaggle
Disney+ Hotstar
Home
External Internship Application
Status
Home
Profile
Pending Applications
Microsoft
Machine Learning
Microsoft
Start Date: 20th December, 2020
Duration: 8 weeks
Stipend: 25,000/-
VIEW RESPONSES
VIEW DETAILS
Cisco
App Development Intern
CISCO
Start Date: 11 January, 2021
Duration: 12 weeks
Stipend: 60,000
VIEW RESPONSES
VIEW DETAILS
Google
Data science Intern
Google
Start Date: March 16th, 2021
Duration: 8 weeks
Stipend: 90,000
VIEW RESPONSES
VIEW DETAILS
Facebook
Research Intern
IBM
Robotics and IoT Intern
Apple
Financial Analyst
ENG
01:19
06-12-2020


InternshipManagementSystem
localhost:4200/teacher-status
Incognito
Update
Gmail
YouTube
Maps
WhatsApp
GitHub
Discord
Prime Video
Google Meet
Kaggle
Disney+ Hotstar
Home
External Internship Application
Status
Home
Profile
Pending Applications
Search Filter
ID
Student Name
Company Name
1 Asher Apple View details
2 Olivia Microsoft View details
3 Atticus Facebook View details
4 Amelia Amazon View details
5 Jack Tesla View details
6 Charlotte Morgan Stanley View details
7 Theodore Facebook View details
ENG
03:06
06-12-2020


InternshipManagementSystem
Zensar Pune - Google Search
localhost:4200/student-profile;id=C2K18106043
Incognito
Update
Gmail
YouTube
Maps
WhatsApp
GitHub
Discord
Prime Video
Google Meet
Kaggle
Disney+ Hotstar
Home
External Internship Application
Status
Home
Profile
Pending Applications
Profile
Id: C2K18106043
Student Information:
Name: Aditya Jadhav
Roll No: 31332
Attendance: 95%
CGPA: 9.69
Department: Computer
Class: TE
Contact: 841292896
Email: adityadjadhav17@gmail.com
Details
Documents
Log Out
Internship Details:
Scope
Undergrad
Type
External internship
Domain
Data Science
Stipend
70,000
Duration:
Start Date
14-02-2021
End Date
14-04-2021
Company Details:
Name
Zensar
Phone
020 6607 4000
Email
support@zensar.com
Location
Kharadi, Pune
Submit
ENG
03:20
06-12-2020


InternshipManagementSystem
localhost:4200/student-profile;id=C2K18106043
Incognito
Update
Gmail
YouTube
Maps
WhatsApp
GitHub
Discord
Prime Video
Google Meet
Kaggle
Disney+ Hotstar
Home
External Internship Application
Status
Home
Profile
Pending Applications
Profile
Id: C2K18106043
Student Information:
Name: Aditya Jadhav
Roll No: 31332
Attendance: 95%
CGPA: 9.69
Department: Computer
Class: TE
Contact: 841292896
Email: adityadjadhav17@gmail.com
Details
Documents
Log Out
Choose File
Resume.pdf
ENG
03:17
06-12-2020

Source Code

Backend routes:

```
from flaskapi import app, db, bcrypt

from flaskapi.models import User, studentInfo, facultyInfo,
internApplicants, internshipDetails, newInternships

from flask_login import login_user, current_user, logout_user,
login_required

from flask import request, jsonify, abort, send_from_directory

from sqlalchemy import and_, or_

from email_validator import validate_email, EmailNotValidError

from datetime import datetime

import json

import re


# db.drop_all()

# db.create_all()


@app.route('/api/complete-profile', methods=['POST'])

def apicompleteprofile():

    if current_user.is_authenticated:

        return jsonify({'logged_in': 'True', 'message': 'Logout to
register a new user'}), 401

    id = request.json.get('id')

    rollNumber = request.json.get('rollNumber')

    f_name = request.json.get('f_name')

    l_name = request.json.get('l_name')

    email = request.json.get('email')

    branch = request.json.get('branch')
```

```
    attendance = request.json.get('attendance')
    CGPA = request.json.get('CGPA')
    year = request.json.get('year')
    division = request.json.get('division')

    regex = re.compile('[@_!#$%^&*()<>?/\|}{~:]')

    if f_name is None or l_name is None or email is None or branch is None or CGPA is None or year is None or division is None:
        return jsonify({'message': 'Fields cannot be blank!'}), 400  # missing arguments
    if(any(chr.isdigit() for chr in f_name) or not(regex.search(f_name) == None)):
        return jsonify({'message': 'Invalid name'}), 400
    if(any(chr.isdigit() for chr in l_name) or not(regex.search(l_name) == None)):
        return jsonify({'message': 'Invalid name'}), 400
    if (type(CGPA) != float and type(CGPA) != int) or CGPA < 0.0:
        return jsonify({'message': 'Invalid CGPA'}), 400
    if (type(attendance) != float and type(attendance) != int) or attendance < 0.0 or attendance > 100.0:
        return jsonify({'message': 'Invalid attendance'}), 400
    if(not (year in ('FE', 'SE', 'TE', 'BE', 'PG'))):
        return jsonify({'message': 'Invalid year'}), 400
    if(type(division) != int) or division < 0 or division > 12:
        return jsonify({'message': 'Invalid division'}), 400
    bool_result = validate_email(email)
    if bool_result is False:
        return jsonify({'message': 'Invalid email'}), 400

    if studentInfo.query.filter_by(rollNumber = rollNumber).first():
```

```
        return jsonify({'registered': 'False', 'message': 'Roll number
is already registered'}), 400

    if studentInfo.query.filter_by(email=email).first():

        return jsonify({'registered': 'False', 'message': 'Email
exists'}), 400




    student = studentInfo(id = id, rollNumber = rollNumber, f_name =
f_name, l_name = l_name, CGPA = CGPA, attendance = attendance, year =
year, division = division, branch = branch, email = email)

    db.session.add(student)

    db.session.commit()



    login_user(user, remember=True)

    return jsonify({'registered': 'True', 'message': 'Account Created',
'id':current_user.id}), 201


@app.route('/api/registerstudent', methods=['POST'])

def apiregisterstudent():

    if current_user.is_authenticated:

        return jsonify({'logged_in': 'True', 'message': 'Logout to
register a new user'}), 401

    id = request.json.get('id')

    password = request.json.get('password')

    regex = re.compile('[@_!#$%^&*()<>?/\|}{~:]')



    if id is None or password is None:

        return jsonify({'message': 'Fields cannot be blank!'}), 400  #
missing arguments

    if len(id) < 8 or len(id) > 13:

        return jsonify({'message': 'Invalid id'}), 400
```

```
    if User.query.filter_by(id=id).first():
        return jsonify({'registered': 'False', 'message': 'id is
already registered'}), 400

    pw_hash = bcrypt.generate_password_hash(password).decode('utf-8')
    user = User(id = id, password = pw_hash, role = 0)
    db.session.add(user)
    db.session.commit()

    return jsonify({'registered': 'True', 'message': 'Account
Created'}), 201

@app.route('/api/studentlogin', methods=['POST','GET'])
def apistudentlogin():
    if request.method == 'GET':
        if current_user.is_authenticated:
            return jsonify({'logged_in': 'True', 'message': 'User was
Logged in Already', 'id':current_user.id}), 200
        else:
            return jsonify({'error': 'Invalid Request'}), 401
    if current_user.is_authenticated:
        return jsonify({'logged_in': 'True', 'message': 'User was
logged in Already'}), 200

    id = request.json.get('id')
    password = request.json.get('password')
    user = User.query.filter_by(id=id).first()
    if user and bcrypt.check_password_hash(user.password, password) and
user.role == 0:
        login_user(user, remember=True)
```

```
        return jsonify({'logged_in': 'True', 'message': 'User logged
in', 'id': current_user.id}), 200
    else:
        return jsonify({'logged_in': 'False', 'message': 'Invalid
credentials'}), 400


@app.route('/api/studentviewinternships', methods=['GET'])
def studentviewinternships():
    if current_user.is_authenticated and current_user.role == 0:
        student =
studentInfo.query.filter_by(id=current_user.id).first()
        internships = newInternships.query.filter_by(department =
student.branch).all()
        interns = []
        for internship in internships:
            interns.append({ 'id': internship.internshipID,
'teacherID': internship.teacherID, 'position': internship.position,
'company': internship.company, 'stipend': internship.stipend,
'duration': internship.duration, 'startDate': internship.startDate,
'endDate': internship.endDate, 'description': internship.description,
'branch': internship.department })
        return jsonify({'internships': interns}), 200


    return jsonify({ 'error': 'Invalid request' }), 400


@app.route('/api/studentviewinternshipsdetails', methods=['GET'])
def viewstudentinternships():
    if current_user.is_authenticated and current_user.role == 0:
        internshipID = request.json.get('internshipID')
        internship = newInternships.query.filter_by(internshipID =
internshipID).first()
        interns = []
```

```
        interns.append({ 'id': internship.internshipID, 'teacherID':
internship.teacherID, 'position': internship.position, 'company':
internship.company, 'stipend': internship.stipend, 'duration':
internship.duration, 'startDate': internship.startDate, 'endDate':
internship.endDate, 'description': internship.description, 'branch':
internship.department })
        return jsonify({'internships': interns}), 200


    return jsonify({ 'error': 'Invalid request' }), 400


@app.route('/api/studentapply', methods=['POST'])
def studentapply():
    if current_user.is_authenticated and current_user.role == 0:
        internshipID = request.form.get('internshipID')
        internship = newInternships.query.filter_by(internshipID =
internshipID).first()
        checkIfApplied = internApplicants.query.filter_by(internshipID
= internshipID, studentID = current_user.id).first()
        if(checkIfApplied):
            return jsonify({'error': 'Internship already applied
for!'}), 400
        file = request.files['file']
        filetype = file.filename.rsplit('.', 1)[1].lower()
        filename = current_user.id + "_resume_" + str(internshipID) +
"." + filetype
        destination="/".join([app.config['UPLOAD_FOLDER'], filename])
        file.save(destination)
        studentApplicant = internApplicants(teacherID =
internship.teacherID, internshipID = internshipID, studentID =
current_user.id, status = 0)
        db.session.add(studentApplicant)
        db.session.commit()
```

```
        return jsonify({ 'applicantRegistered': 'True', 'message':
'Application sent.'}), 200


    return jsonify({'error': 'Invalid request'}), 400


@app.route('/api/viewstudentprofile', methods=['GET'])
def studentviewprofile():
    if current_user.is_authenticated and current_user.role == 0:
        student = studentInfo.query.filter_by(id =
current_user.id).first()
        return jsonify({ 'ID': student.id, 'rollNumber':
student.rollNumber, 'f_name': student.f_name, 'l_name': student.l_name,
'CGPA': student.CGPA, 'attendance': student.attendance, 'year':
student.year, 'division': student.division, 'branch': student.branch,
'email': student.email}), 200


    return jsonify({'error': 'Invalid request'}), 400




@app.route('/api/studentviewstatus', methods=['GET'])
def studentviewstatus():
    if current_user.is_authenticated and current_user.role == 0:
        externalApplications =
internshipDetails.query.filter_by(studentID = current_user.id).all()
        collegeInternshipApplications =
internApplicants.query.filter_by(studentID = current_user.id).all()
        allApplications = []


        collegeApplicantsStatus = {
            0: 'Not viewed',
            1: 'Viewed',
            2: 'Approved',
```

```
            3: 'Rejected'

        }

        for internship in collegeInternshipApplications:

            currentInternship =
newInternships.query.filter_by(internshipID =
internship.internshipID).first()

            allApplications.append({'id': internship.internshipID,
'name': currentInternship.company, 'type': 'College internship
application', 'status': collegeApplicantsStatus.get(internship.status)
})



        for extInternship in externalApplications:

            allApplications.append({'id': extInternship.formID, 'name':
extInternship.company, 'type': 'External internship approval
application', 'status':
collegeApplicantsStatus.get(extInternship.status)})


        return jsonify({ 'internshipStatuses': allApplications }), 200


    return jsonify({ 'error': 'Invalid request'}), 400


@app.route('/api/requestapproval', methods=['POST'])
def requestapproval():
    if current_user.is_authenticated and current_user.role == 0:
        studentID = current_user.id
        company = request.form.get("company")
        stipend = request.form.get("stipend")
        duration = request.form.get("duration")
        startDate = request.form.get("startDate")
        endDate = request.form.get("endDate")
        startDate = datetime.strptime(startDate, '%Y-%m-%d')
        endDate = datetime.strptime(endDate, '%Y-%m-%d')
```

```
        additionalDetails = request.form.get("additionalDetails")

        checkIfApplied = internshipDetails.query.filter_by(studentID =
current_user.id, company = company).first()

        if(checkIfApplied):

            return jsonify({'error': 'Internship approval already
requested for this company.'}), 400

        file = request.files['file']

        filetype = file.filename.rsplit('.', 1)[1].lower()

        filename = current_user.id + "_offer_letter_" + company + "." +
filetype

        destination="/".join([app.config['UPLOAD_FOLDER'], filename])

        file.save(destination)

        internship_dets = internshipDetails(studentID=studentID,
company=company,stipend=stipend, duration=duration,
startDate=startDate, endDate=endDate,
additionalDetails=additionalDetails, status=0)

        db.session.add(internship_dets)

        db.session.commit()



        return jsonify({'message': 'Approval request sent.'}), 201


    return jsonify({'error': 'Invalid request'}), 400


@app.route('/api/registerfaculty', methods=['POST'])
def apiregister():

    if current_user.is_authenticated:

        return jsonify({'logged_in': 'True', 'message': 'Logout to
register a new user'}), 401

    id = request.json.get('id')

    f_name = request.json.get('f_name')

    l_name = request.json.get('l_name')
```

```
    email = request.json.get('email')
    password = request.json.get('password')
    department = request.json.get('department')
    securityClearance = request.json.get('securityClearance')
    regex = re.compile('[@_!#$%^&*()<>?/\|}{~:]')


    if id is None or f_name is None or l_name is None or email is None
or password is None or department is None or securityClearance is None:
        return jsonify({'message': 'Fields cannot be blank!'}), 400  #
missing arguments
    if len(id) < 8 or len(id) > 13:
        return jsonify({'message': 'Invalid id'}), 400
    if(any(chr.isdigit() for chr in f_name) or not(regex.search(f_name)
== None)):
        return jsonify({'message': 'Invalid name'}), 400
    if(any(chr.isdigit() for chr in l_name) or not(regex.search(l_name)
== None)):
        return jsonify({'message': 'Invalid name'}), 400
    bool_result = validate_email(email)
    if bool_result is False:
        return jsonify({'message': 'Invalid email'}), 400
    if User.query.filter_by(id=id).first():
        return jsonify({'registered': 'False', 'message': 'id is
already registered'}), 400
    if facultyInfo.query.filter_by(email=email).first():
        return jsonify({'registered': 'False', 'message': 'Email
exists'}), 400


    pw_hash = bcrypt.generate_password_hash(password).decode('utf-8')
    user = User(id = id, password = pw_hash, role = 1)
    db.session.add(user)
```

```
    db.session.commit()

    faculty = facultyInfo(id = id, f_name = f_name, l_name = l_name,
email = email, department = department, securityClearance =
securityClearance)

    db.session.add(faculty)

    db.session.commit()



    login_user(user, remember=True)

    return jsonify({'registered': 'True', 'message': 'Account
Created','id':current_user.id}), 201



@app.route('/api/facultylogin', methods=['POST','GET'])

def apilogin():

    if request.method == 'GET':

        if current_user.is_authenticated:

            return jsonify({'logged_in': 'True', 'message': 'User was
Logged in Already','id':current_user.id}), 200

        else:

            return jsonify({'error': 'Invalid Request'}), 401

    if current_user.is_authenticated:

        return jsonify({'logged_in': 'True', 'message': 'User was
logged in Already'}), 200



    id = request.json.get('id')

    password = request.json.get('password')

    user = User.query.filter_by(id=id).first()

    print(user)

    if user and bcrypt.check_password_hash(user.password, password) and
user.role != 0:

        login_user(user, remember=True)
```

```
        return jsonify({'logged_in': 'True', 'message': 'User logged
in', 'id':current_user.id}), 200
    else:
        return jsonify({'logged_in': 'False', 'message': 'Invalid
credentials'}), 400


@app.route('/api/logout', methods=['GET', 'POST'])
def apilogout():
    if current_user.is_authenticated:
        logout_user()
        return jsonify({'logged_out': 'True', 'message': 'User logged
out'}), 200
    else:
        return jsonify({'error': 'Invalid Request'}), 401


@app.route('/api/addinternship', methods=['POST'])
def apiaddinternship():
    if current_user.is_authenticated and current_user.role != 0:
        faculty =
facultyInfo.query.filter_by(id=current_user.id).first()
        position = request.json.get('position')
        company = request.json.get('company')
        stipend = request.json.get('stipend')
        duration = request.json.get('duration')
        startDate = request.json.get('startDate')
        endDate = request.json.get('endDate')
        startDate = datetime.strptime(startDate, '%Y-%m-%d')
        endDate = datetime.strptime(endDate, '%Y-%m-%d')
        description = request.json.get('description')
        branch = request.json.get('branch')
```

```
        newintern = newInternships(teacherID = current_user.id,
position = position, company = company, stipend = stipend, duration =
duration, startDate = startDate, endDate = endDate, description =
description, department = branch)
        db.session.add(newintern)
        db.session.commit()
        return jsonify({ 'message': 'Internship added'}), 201

    return jsonify({ 'error': 'Invalid request' }), 400

@app.route('/api/view-added-internships', methods = ['GET'])
def apigetinternships():
    if current_user.is_authenticated and current_user.role != 0:
        internships = newInternships.query.filter_by(teacherID =
current_user.id).all()
        interns = []
        for internship in internships:
            interns.append({ 'id': internship.internshipID, 'position':
internship.position, 'company': internship.company, 'stipend':
internship.stipend, 'duration': internship.duration, 'startDate':
internship.startDate, 'endDate': internship.endDate, 'description':
internship.description, 'branch': internship.department })

        return jsonify({'internships': interns}), 200

    return jsonify({ 'error': 'Invalid request' }), 400

@app.route('/api/all-internships', methods = ['GET'])
def apigetallinternships():
    if current_user.is_authenticated and current_user.role != 0:
        internships = newInternships.query.all()
        interns = []
```

```
        for internship in internships:
            interns.append({ 'id': internship.internshipID, 'position':
internship.position, 'company': internship.company, 'stipend':
internship.stipend, 'duration': internship.duration, 'startDate':
internship.startDate, 'endDate': internship.endDate, 'description':
internship.description, 'branch': internship.department })

        return jsonify({'internships': interns}), 200

    return jsonify({ 'error': 'Invalid request' }), 400

@app.route('/api/view-responses', methods = ['GET'])
def apiviewresponses():
    if current_user.is_authenticated and current_user.role != 0:
        internshipID = request.json.get('internshipID')
        applicants = internApplicants.query.filter_by(internshipID =
internshipID).all()
        studentResponses = []
        for applicant in applicants:
            student = studentInfo.query.filter_by(id =
applicant.studentID).first()
            studentResponses.append({ 'id': student.id, 'rollNumber':
student.rollNumber, 'name': student.f_name + ' ' + student.l_name,
'class': student.year + '-' + str(student.division)})

        return jsonify({ 'studentResponses': studentResponses }), 200

    return jsonify({ 'error': 'Invalid request' }), 400

@app.route('/api/view-applicant-profile', methods = ['GET'])
def apiviewapplicantprofile():
    if current_user.is_authenticated and current_user.role != 0:
```

```
        studentID = request.json.get('studentID')
        student = studentInfo.query.filter_by(id = studentID)
        try:
            return jsonify({ 'id': student.id, 'rollNumber':
student.rollNumber, 'f_name': student.f_name, 'l_name': student.l_name,
'year': student.year, 'CGPA': student.CGPA, 'attendance':
student.attendance, 'branch': student.branch, 'email': student.email
}), 200
        except:
            return jsonify({ 'error': 'Student not found' }), 400

    return jsonify({ 'error': 'Invalid request' }), 400

@app.route('/api/view-status-teacher', methods = ['GET'])
def apiviewapprovalrequests():
    if current_user.is_authenticated and current_user.role != 0:
        applications =
db.session.query(internshipDetails).filter(internshipDetails.status <
2)
        applicationDetails = []
        for application in applications:
            student = studentInfo.query.filter_by(id =
application.studentID).first()
            applicationDetails.append({ 'id': application.formID,
'name': student.f_name + ' ' + student.l_name, 'company':
application.company, 'typeOfApplication': 'External internship
approval'})

        return jsonify({'applications': applicationDetails})

    return jsonify({ 'error': 'Invalid request' }), 400
```

```
@app.route('/api/view-approval-request-details', methods = ['GET'])
def apiviewappdetails():
    if current_user.is_authenticated and current_user.role != 0:
        formID = request.json.get('id')
        application = internshipDetails.query.filter_by(formID =
formID).first()
        filename = current_user.id + "_offer_letter_" +
application.company + ".pdf"
        student = studentInfo.query.filter_by(id =
application.studentID).first()
        return jsonify({'id': application.formID, 'name':
student.f_name + student.l_name, 'rollNumber': student.rollNumber,
'CGPA': student.CGPA, 'attendance': student.attendance, 'company':
application.company, 'startDate': application.startDate, 'endDate':
application.endDate, 'duration': application.duration, 'stipend':
application.stipend, 'additionalDetails':
application.additionalDetails}), 200

    return jsonify({ 'error': 'Invalid request' }), 400
```

Testing Document

Student user:

Description	Input	Output	Result

Sign up as a registered student	{
	“id”: “C2K18106045”,
	“password”: “walt12”
	“typeOfUser”: “Student”,
	}
		Code: 201, created.
		{ “Message”: “Student registered.” }
			Pass

Login a registered student	{ “id”: C2K18106045 “password”: “walt12”	Code: 200, Message: Logged in	Pass

	}		
Apply for an internal internship	Resume: <attach-file>	Code: 200	Pass
View status of applications	——	Code: 200, Body: { Application status JSON data }	Pass
Request approval for external internship	Internship details (company, stipend, duration) Offer letter: <attach-file>	Code: 200	Pass

Modify student profile	{
	“rollNumber”: 31391
	}
		Code: 200,
		{ “Message”: “Profile details updated.” }
			Pass

Teacher:

Description	**Input**	**Output**	**Result**
Login a registered teacher	ID: T2K18106045 Password: sample12*	Code: 200, Message: Logged in	Pass
Add internal college internship	Internship details (company, stipend, duration)	Code: 200	Pass

View responses for internal college internship	{
	“internshipID”: 2
	}
		200, { JSON data of students who applied }	Pass

Approve/ reject student external internship	{
	“applicationID”: 1,
	“status”: “Approved”
	}
		Code: 200	Pass

Generate student internship reports (monthly/quarterly/ annually)	——	Code: 200, { JSON data of internship reports }	Pass
Save student internship reports	——	Code: 200, CSV file downloaded	Pass

Conclusion

In most colleges, internship approval is a lengthy manual process for students and teachers alike. In this report, we discussed the development of a system that automates this process. We looked at the various flows for all users of the system, the tables we needed, and the programming languages used. Our initial objectives were simple – provide an integrated platform to streamline and automate the internship approval process, as well as providing insights and statistics to the college / university, that could not normally be obtained. Both objectives were met.

This lab introduced us to the basics of database management systems, the importance of constructing ERDs, and the need for normalization of data. We implemented all these concepts in the development of this system. At a larger scale, we feel this project can be deployed in colleges all over to streamline the internship process, and we are grateful for the help, guidance and support we received during its development.

References

[1] The importance of data modelling - https://www.guru99.com/data-modelling-conceptual-logical.html

[2] Creating our ERD – https://www.lucidchart.com

[3] Creating our use case diagram - https://creately.com/

[4] Data dictionaries and how to make one - https://www.youtube.com/watch?v=kH0bcw9P2Lc

[5] Normal forms in DBMS - https://www.javatpoint.com/dbms-normalization

[6] How to write a clear conclusion to a report - https://my.ece.utah.edu/~ece1270/CLEAR_10_Conclusions.pdf