

Computing Studies & Information Systems

"Applied Research Project"

Fall 2022: CSIS 4495 – Section 050

Progress Report 2

"TeamSavvy"



by:

- | | |
|-----------------------------|------------------------------|
| • Name: Jasmine Kaur | Student ID: 300345933 |
| • Name: Jaspal Singh | Student ID: 300345938 |
| • Name: Harman Lal | Student ID: 300343291 |
| • Name: Joling Weng | Student ID: 300335548 |



Statement of the Problem:

The Employee Management System helps create an environment that motivates communication and keeps employees and managers working together to meet company goals. It also helps the company to manage the information and activities of employees.

The issue with the existing system is to access data about employee projects, leaves, payroll, and jobs applied internally, for accessing these details an individual has to use different web platforms.

Our system proposes that a person can access the details for projects, work history, team hierarchy, peers, timesheets, leaves and payroll information on one platform. The managers and company owners can access the records of employees and also check the resume of the candidates who applied for jobs on their company website with an improved user interface for a better user experience.

Significance of the study

Who would benefit from this project and how?

Employee Engagement: Our platform will provide employees to add their details, apply for leaves, and check their work history and payroll details on one platform. This will save time for an individual to concentrate on tasks for company growth. Employees can track their growth and improve their working efficiency.

Biometric: We will create an android app to get user biometric timestamp and will store it in the database. Employee can use his biometric to clock-in and clock-out of the system.

Create Dashboard: Our project will provide the dashboard of collective data of employees. A manager or company owner can perform descriptive analysis on employee's data; with respect to programming languages, projects, leaves (sick leave, casual leave) in specific period and, salary. **We will give different graph options to select from and, can create multiple graphs to visualize data and show the dashboard on preview mode and can download it.** The HR at a company can do analytics; for example, number of employees have knowledge of programming language, salary of each employee in specific project. We will represent this information in a graphical format.

Create Task: The employee can fill his hours he is working on a particular task and then change the status of task to in-progress, after completing the task employee can change the task status to finish. The notification will be sent to the manager that this task has been completed. Manager can track the progress of the tasks assigned to team members.



Improve communication: Our platform provides the information of each employee to others by search option that will let them to stay connected, and an easy path for new joiners to get to know their peers. All in all, improve the connection which will reduce the problem of communication gap within an organization.

Users:

Employee:

- An employee will have access to a calendar where he can see clock-in and clock-out for a day
- Check total working hours
- Apply for leaves
- Check his payroll details (Read only),
- Get information of other employee working in an organization for communication
- Can get the team hierarchy
- Check the number and details of projects he is working on
- Employee will have the profile page where he will have his information saved or can update if required
- Can apply on internal jobs
- Apply for resignation

Admin (HR or Manager):

- An admin will have all the functionality that employee has
- Able to see information of employees in his team
- He can perform CRUD operations on them
- Check payroll info and can give increment to an employee
- Admin will have details of project's budget and deadline
- Can assign project to employee
- Modify involvement of employees in different projects,
- Can approve leaves of an employee
- Check timesheets
- Create internal job postings for employees.

Super admin (Company Owner): A super admin can perform CRUD operations on admins and employees.

Project Implementations:

What have we done so far ?

Test Cases and screenshot: Testing the API gives the following results:



Checking the Employee Database:

In this we enter the employee id which will correspond to the employee details associated to the database. As an output of this we get the response from the API with desired user details.

GET /api/Employee/{employeeId}

Parameters

Name	Description
employeeId * required	integer(\$int32) (path)
1	

Responses

Curl

```
curl -X 'GET' \
  'https://localhost:44362/api/Employee/1' \
  -H 'accept: application/json'
```

Request URL

<https://localhost:44362/api/Employee/1>

Server response

Code Details

200 Response body

```
{
  "success": true,
  "response": {
    "employeeId": 1,
    "employeeFirstName": "Laspal",
    "employeeLastName": "Singh",
    "dateOfBirth": "1995-10-09",
    "hireDate": "2015-09-08",
    "phone": "+046785242",
    "email": "laspalsingh@gmail.com",
    "extension": "1000",
    "bankAccount": "600678563",
    "employeeImage": null,
    "password": null,
    "bankName": "SCOTIA",
    "bankAcc": "11005",
    "role": {
      "statusId": 1,
      "statusType": "Active"
    },
    "role": [
      {
        "roleId": 2,
        "roleType": "Admin"
      }
    ],
    "department": [
      {
        "departmentId": 1,
        "departmentName": "Finance"
      }
    ]
  }
}
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Mon, 07 Nov 2022 19:01:32 GMT
server: Microsoft-IIS/10.0
x-powered-by: ASP.NET
```

Responses

Code Description Links

200 SUCCESS No links



To add a new Employee :

When HR enter the details of a new employee. The API will save the data from frontend to database. This part of the API needs fixation as so far we're not able to add the Employee. It will generate an error stating that response status is 400.

The screenshot shows a REST client interface with a green header bar containing "POST /api/Employee/addEmployee". Below it is a "Parameters" section with a "Cancel" and "Reset" button. A "Request body" section is set to "application/json" and contains a JSON object representing an employee. The JSON object includes fields like employeeId, employeeFirstName, employeeLastName, dateOfBirth, hireDate, phone, email, extension, department, employeeImage, password, address, backOffice, and status. The status field is defined with "statusId": 1 and "statusType": "Active". The request body also includes a "role" field with "mn1=Td". At the bottom are "Execute" and "Clear" buttons. Below the main area is a "Responses" section.

Request URL: <https://localhost:44362/api/Employee/addEmployee>

Server response:

Code	Details
400	Error: response status is 400

Response body:

```
{ "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1", "title": "One or more validation errors occurred.", "status": 400, "traceId": "44d4865f-4d5bcdc46e256c8e.", "errors": { "$employeeImage": [ "The JSON value could not be converted to System.Byte[]. Path: $.employeeImage | LineNumber: 10 | BytePositionInLine: 27." ] } }
```

Response headers:

```
access-control-allow-origin: *  
content-type: application/json; charset=utf-8  
date: Mon, 07 Nov 2022 19:14:55 GMT  
server: Microsoft-IIS/10.0  
x-powered-by: ASP.NET
```

Due to this **Deletion of an Employee** cannot be done.



Jobs :

The job API retrieves the data from database that is related to the Jobs posted in the company.

Job

GET /api/Job

Parameters

No parameters

Responses

Curl

```
curl -X 'GET' \
  'https://localhost:44362/api/Job' \
  -H 'accept: application/json'
```

Request URL

```
https://localhost:44362/api/Job
```

Server response

Code Details

200 Response body

```
{
  "success": true,
  "response": [
    {
      "jobId": 1,
      "jobPosition": "Project Manager",
      "salary": "40000",
      "details": "Project Manager"
    }
  ]
}
```

Code Details

200 Response body

```
{
  "success": true,
  "response": [
    {
      "jobId": 1,
      "jobPosition": "Project Manager",
      "salary": "40000",
      "details": "Project Manager",
      "responsibilities": "Graduation Level",
      "createdOn": "2022-09-08",
      "deadline": "2022-09-08",
      "isDelete": false,
      "jobSkills": []
    },
    {
      "jobId": 2,
      "jobPosition": "Data Scientist",
      "salary": "80000",
      "details": "Data Science Projects",
      "responsibilities": "Masters in Data Science",
      "createdOn": "2022-07-10",
      "deadline": "2022-09-08",
      "isDelete": false,
      "jobSkills": [
        {
          "jobSkillId": 4,
          "jobId": 2,
          "details": "Machine Learning"
        }
      ]
    }
  ]
}
```

Response headers

```
content-type: application/json; charset=utf-8
date: Mon, 07 Nov 2022 19:27:43 GMT
server: Microsoft-IIS/10.0
x-powered-by: ASP.NET
```

Responses

Code Description

200 Success

Links

No links



Add a Job:

While adding a new job, generate a new job posting with required attributes. Similar to adding the Employee, this part gives an error code 400 as well.

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** /api/Job/addJob
- Parameters:** No parameters
- Request body:** application/json
- Body Content:** A JSON object representing a job posting. The JSON includes fields like jobId, jobPosition, salary, deadline, responsibilities, createdOn, deadline, isDelete, and jobSkills. One of the skills in the jobSkills array has its id field set to 0, which is highlighted in red.
- Buttons:** Execute (blue button) and Clear (white button).
- Responses:** A section showing curl command examples and a request URL.
- Curl:** curl -X 'POST' '\n https://localhost:44362/api/Job/addJob' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
 "jobId": 0,
 "jobPosition": "Accountant",
 "salary": "600000",
 "deadline": "Deals with the Assets",
 "responsibilities": "Account Management",
 "createdOn": "2022-11-09",
 "deadline": "2022-12-09",
 "isDelete": true,
 "jobSkills": [
 {
 "jobSkillId": 4,
 "jobId": 0,
 "skillId": 2,
 "skills": [
 {
 "skillId": 2,
 "skillName": "Python"
 }
]
 }
]
}'
- Request URL:** https://localhost:44362/api/Job/addJob
- Server response:** Code 404, Details: Error: response status is 404

Therefore, the deletion of a job will not proceed as well.



Payroll:

Payroll API gets the data of monthly salary associated to each employee. And holds the record of all the increments, deduction, Sick pay, Vacation pay, Number of hours an employee has worked on monthly bases. Calculating all these attributes gives the actual payment associated to each employee.

Payroll

GET /api/Payroll

Parameters

No parameters

Responses

Curl

```
curl -X GET 'https://localhost:44362/api/Payroll' \
-H 'accept: application/json'
```

Request URL

```
https://localhost:44362/api/Payroll
```

Server response

Code Details

200 Response body

```
{
  "success": true,
  "response": [
    {
      "payrollId": 1,
      "employeeId": 1,
      "payDate": "2022-10-07T00:00:00",
      "payType": "Monthly"
    }
  ]
}
```

200 Response body

```
{
  "success": true,
  "response": [
    {
      "payrollId": 1,
      "employeeId": 1,
      "payDate": "2022-10-07T00:00:00",
      "payType": "Monthly",
      "totalHours": 160,
      "earning": 4167,
      "netpay": 4167,
      "paySick": 100,
      "payVacation": 100,
      "deduction": 2500,
      "salaryId": 1
    },
    {
      "payrollId": 2,
      "employeeId": 2,
      "payDate": "2022-11-07T00:00:00",
      "payType": "Monthly",
      "totalHours": 160,
      "earning": 4167,
      "netpay": 50000,
      "payTid": 4167,
      "payVacation": 100
    }
  ]
}
```

Response headers

```
content-type: application/json; charset=utf-8
date: Mon, 07 Nov 2022 19:41:31 GMT
server: Microsoft-IIS/10.0
x-powered-by: ASP.NET
```

Responses

Code Description Links

200 Success No links

Execute Clear



Payroll List with respect to Employee ID :

This API will use employee id and show all the information of payroll receipts that employee has received throughout the Year.

GET /api/Payroll/list/employeeId/{employeeId}

Parameters

Name	Description
employeeId * required	integer(\$int32) (path)
1	

Cancel

Responses

Curl

```
curl -X 'GET' \
  'https://localhost:44362/api/Payroll/list/employeeId/1' \
  -H 'accept: application/json'
```

Request URL

```
https://localhost:44362/api/Payroll/list/employeeId/1
```

Server response

Code Details

200 Response body

```
{
  "success": true,
  "response": [
    {
      "payrollId": 1,
      "employeeId": 1,
      "payDate": "2022-10-08T00:00:00",
      "payType": "Monthly",
      "totalHours": 168,
      "earning": "4167",
      "netpay": "56000",
      "payTzd": "4167",
      "payVacation": "100",
      "paySick": "100",
      "deduction": "2500",
      "salaryId": 1
    },
    {
      "payrollId": 9,
      "employeeId": 1,
      "payDate": "2022-11-08T00:00:00",
      "payType": "Monthly",
      "totalHours": 168,
      "earning": "4167",
      "netpay": "56000",
      "payTzd": "8334",
      "payVacation": "100",
      "paySick": "100",
      "deduction": "2500",
      "salaryId": 1
    }
  ]
}
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Mon, 07 Nov 2022 19:46:20 GMT
server: Microsoft-IIS/10.0
x-powered-by: ASP.NET
```

Responses

Code Description Links

200 SUCCESS No links



Projects:

This will display all the projects that are undergoing in the company with detailed information associated to Employee Id and Project Id.

Projects

GET /api/Projects

Parameters

No parameters

Responses

Curl

```
curl -X 'GET' \
'https://localhost:44362/api/Projects' \
-H 'accept: application/json'
```

Request URL

<https://localhost:44362/api/Projects>

Server response

Code Details

200 Response body

```
{
  "success": true,
  "response": [
    {
      "projectId": 1,
      "employeeId": 1,
      "employeeProjectId": 1,
      "projectName": "Weather Forecasting",
      "projectStartDate": "2022-10-10T00:00:00",
      "projectEndDate": "2023-10-10T00:00:00",
      "projectBudget": "15000"
    }
  ]
}
```

200 Response body

```
{
  "success": true,
  "response": [
    {
      "projectId": 2,
      "employeeId": 2,
      "employeeProjectId": 2,
      "projectName": "Weather Forecasting",
      "projectStartDate": "2022-10-10T00:00:00",
      "projectEndDate": "2023-10-10T00:00:00",
      "projectBudget": "15000"
    }
  ]
}
```

Response headers

```
content-type: application/json; charset=utf-8
date: Mon, 07 Nov 2022 20:10:19 GMT
server: Microsoft-IIS/10.0
x-powered-by: ASP.NET
```

Responses

Code Description

200 Success

Links

No links



Project with respect to Employee ID:

This API will use employee id and display the projects that are associated to that employee. Managers are assigned more than one projects so this will give the information of all the projects a manager is monitoring over.

GET /api/Projects/employeeId/{employeeId}

Parameters

Name	Description
employeeId * required	integer(\$int32) (path) 9

Responses

Curl

```
curl -X 'GET' \
'https://localhost:44362/api/Projects/employeeId/9' \
-H 'accept: application/json'
```

Request URL

<https://localhost:44362/api/Projects/employeeId/9>

Server response

Code Details

200 Response body

```
{
  "success": true,
  "response": [
    {
      "projectId": 8,
      "employeeId": 9,
      "employeeProjectId": 16,
      "projectName": "Statistical Analysis",
      "projectStartData": "2022-03-04T00:00:00",
      "projectEndData": "2023-03-04T00:00:00",
      "projectBudget": "20000",
      "projectDesc": "Introduction Purpose of this Project,to find out the relationship between population and the square of countries is tried to be understood. This Project includes 129 observations and 38 samples.We tested with 95% confidence interval Step 1:State null and alternate hypotheses Step 2:Select a level of significance Step 3:Identify the test statistic Step 4:Formulate the decision rule Step 5:Take a sample,arrive at a decision Step 1 Hypothesis Test: H0:The average population of countries is equal",
      "totalTaskCount": 10,
      "totalCompletedCount": 8,
      "projectManagerId": 9,
      "projectManagerName": "Jo",
      "projectStatus": "Active",
      "projectLead": "Phoenix",
      "projectTotalEmployees": 4,
      "employeeProjectStatus": true
    },
    {
      "projectId": 9,
      "employeeId": 9,
      "employeeProjectId": 25,
      "projectName": "New Project 3"
    }
  ]
}
```

Download

Response header

```
content-type: application/json; charset=utf-8
date: Mon, 07 Nov 2022 20:19:29 GMT
server: Microsoft-IIS/10.0
x-powered-by: ASP.NET
```

Responses

Code Description

200 Success

Links

No links



Adding a New Project:

This API will get information from Create Project screen and . Since, project Id is primary key this will be added automatically in the Database.

The screenshot shows a REST API testing interface with the following details:

- Method:** POST
- URL:** /api/Projects/addProject
- Parameters:** No parameters
- Request body:** application/json
- Request Body Content:**

```
{
  "projectId": 0,
  "employeeId": 32,
  "employeeProjectId": 11,
  " projectName": "Uber App",
  "projectStartDate": "2025-12-19T20:25:53.859Z",
  "projectEndDate": "2025-12-19T20:25:53.859Z",
  "projectBudget": "600000",
  "projectDesc": "Uber Cab Service",
  "totalTaskCount": 10,
  "totalCompletedCount": 5,
  "projectManagerId": 10,
  "projectManagerName": "Harry",
  "projectClient": "Nipro",
  "projectLead": "Jack",
  "projectTotalEmployees": 4,
  "employeeProjectStatus": true
}
```
- Buttons:** Execute, Clear
- Responses:** Responses tab is collapsed.
- Curl:** curl -X 'POST' '\nhttps://localhost:44362/api/Projects/addProject' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d {
 "projectId": 0,
 "employeeId": 32,
 "employeeProjectId": 11,
 " projectName": "Uber App",
 "projectStartDate": "2025-12-19T20:25:53.859Z",
 "projectEndDate": "2025-12-19T20:25:53.859Z",
 "projectBudget": "600000",
 "projectDesc": "Uber Cab Service",
 "totalTaskCount": 10,
 "totalCompletedCount": 5,
 "projectManagerId": 10,
 "projectManagerName": "Harry",
 "projectClient": "Nipro",
 "projectLead": "Jack",
 "projectTotalEmployees": 4,
 "employeeProjectStatus": true
}
- Request URL:** https://localhost:44362/api/Projects/addProject
- Server response:**

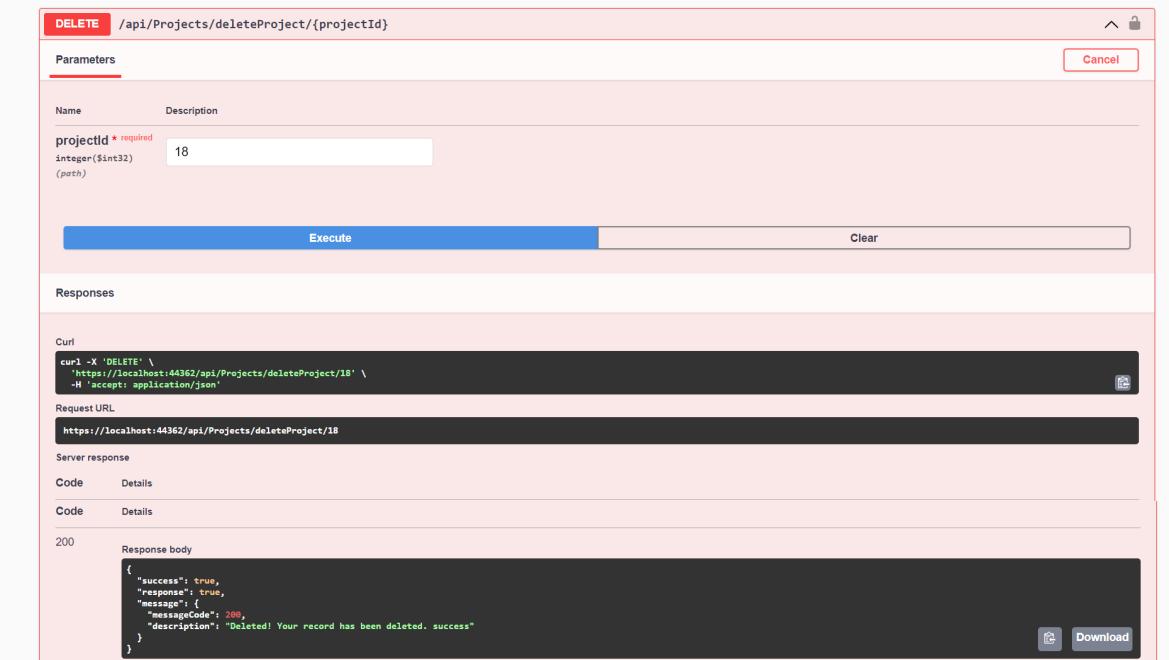
Code	Details
200	Response body

```
{
  "success": true,
  "response": true,
  "message": "Success",
  "statusCode": 200,
  "description": "OK"
}
```



Deleting a Project:

It will ask user to enter the project Id that user wants to delete and it will erase the row associated to that project id in the Database.



The screenshot shows a POSTMAN interface for a DELETE request. The URL is /api/Projects/deleteProject/{projectId}. The method is set to DELETE. In the 'Parameters' section, there is one parameter named 'projectId' with the value '18'. Below the parameters are 'Execute' and 'Clear' buttons. Under 'Responses', there are sections for 'Curl' (containing the curl command), 'Request URL' (containing the URL https://localhost:44362/api/Projects/deleteProject/18), and 'Server response'. The 'Server response' section shows a status code of 200 with a response body containing JSON: { "success": true, "response": true, "message": "Deleted!", "messageCode": 200, "description": "Deleted! Your record has been deleted. success" }. There are 'Copy' and 'Download' buttons next to the response body.

Fetch API & Client side React code

After creating the front-end UI and back-end logic, from this stage we are focusing on connecting the front-end and back-end together. Although different React structures are still used widely, we chose the most popular and on trend one – hook, to manage the data flow. Since we have 21 tables and populating data takes time, the pages may seem empty. However, the data now is mostly dynamic and be able to maintain using CRUD.

Profile Update

- Fetching and putting employee data from/ to API
- Adding skills showing on page as badges
- After clicking on resignation, system will send notification to manager's email

DOUGLAS COLLEGE

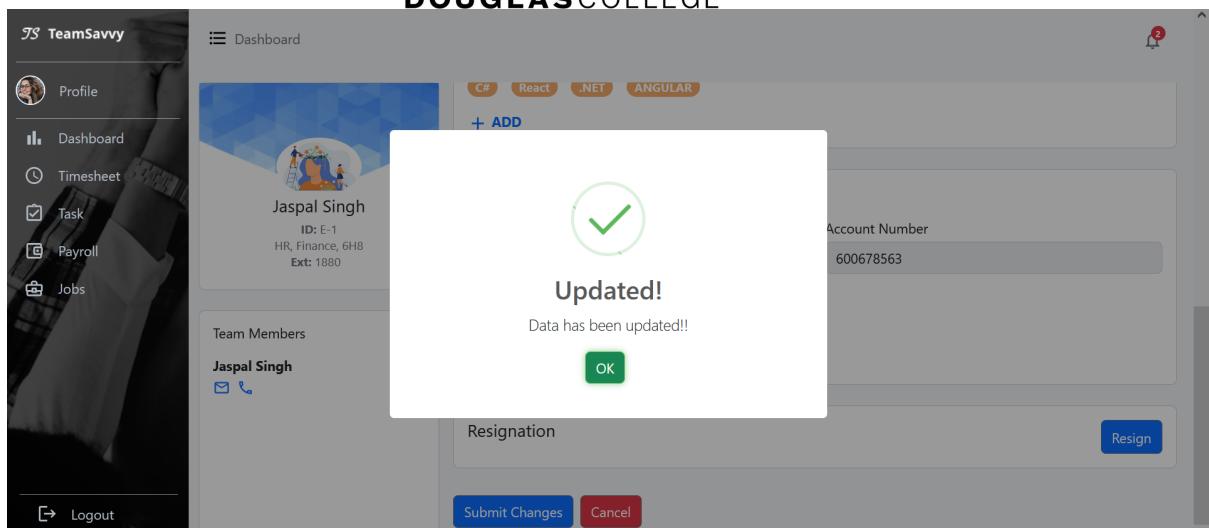


FIGURE 1: EMPLOYEE PROFILE UPDATE TO DATABASE

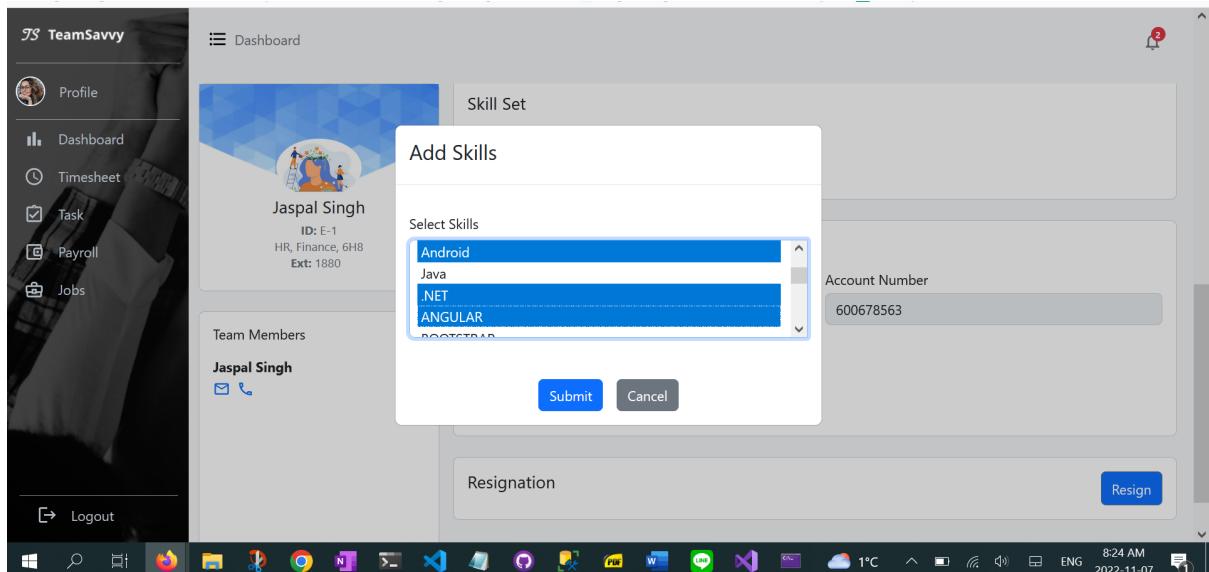


FIGURE 2: MULTIPLE SKILLS CAN BE ADDED AND DELETED

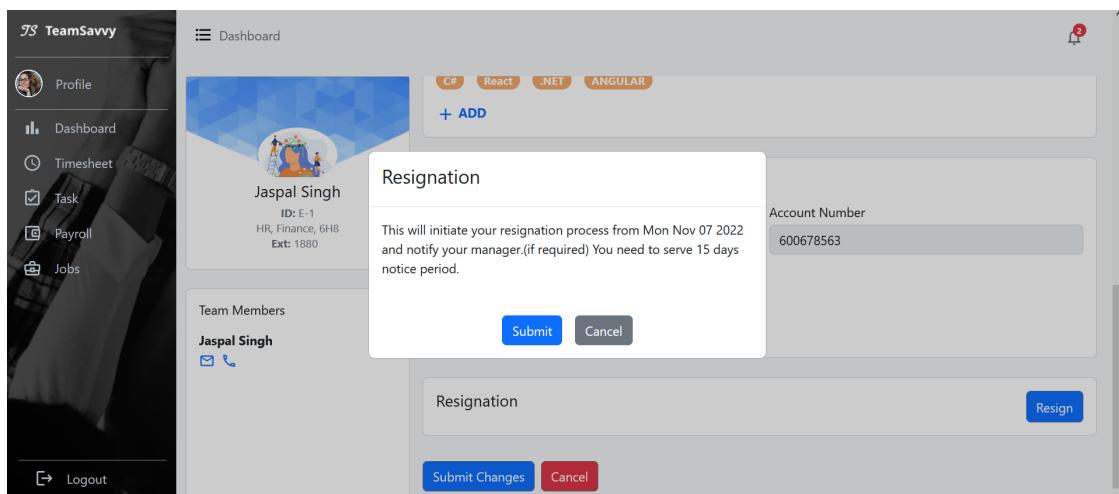


FIGURE 3:SIMPLIFY RESIGNATION PROCESS ALLOWING EMPLOYEE TO SEND AUTO RESIGNATION EMAIL TO MANAGER BY SIMPLE CLICK



Modify and validate task:

- Fetch data from api: get assigned jobs list
- Modal input validation: validate input at run time, preventing modal from closing if data is empty or not correct.
- Put request: update modified task to api using http put request

FIGURE 4: SIMPLE VALIDATION FOR INPUT. MODIFIED TASK WILL BE UPDATED TO DATABASE

Payslips and pay stub

Fetching data from payroll and employee API, using Links to pass prop and render detail in the child page

Pay Date	Pay Type	Total Hours	Earnings	Net Pay(after taxes)	File
09-Sep-2022	Monthly	40	\$5000	\$60000	VIEW FILE
28-FEB-2022	Monthly	80	\$3600	\$3200	VIEW FILE
31-JAN-2022	Monthly	80	\$3600	\$3200	VIEW FILE

FIGURE 5: FETCH PAYSLIPS DATA FOR INDIVIDUALS. PAYSTUB DETAILS WILL BE SHOWN IN VIEW FILE.



The screenshot shows the TeamSavvy application interface. On the left is a sidebar with icons for Profile, Dashboard, Timesheet, Task, Payroll, and Jobs, along with a Logout button. The main area is titled "Dashboard". It displays the following information:

- Employee Details:** Jaspal Singh, Employee#1, Department: Finance, Start Date 07-Nov-2022, End Date
- Address:** 152 st, Vancouver, BC, V6E 9E8
- Organization:** 6H8, Langley, BC, HDK8
- EARNINGS:** Type: Monthly, Total Working days: 80, Amount: \$60000, YTD: \$30000
- PAY DISTRIBUTION:** Type: Electronic bank transfers, Amount: \$60000, Account#: 600678563, Bank: SCOTIA

FIGURE 6: PASSING DATA FROM PAYROLL AND EMPLOYEE API

Jobs

- View change according to the employee role. For example, engineer won't be able to create jobs and see the applied details
- Fetching data from Jobs API, clicking on the view will show the corresponding detail on the right-side card.
- Clicking on Create Job shows the job creating card. The post request here hasn't implemented yet.
- Clicking on the applied button shows the employee applied for this job. The detail link will then show the employee's profile.

The screenshot shows the HR view of the job page. The sidebar on the left is identical to Figure 6. The main area shows a job listing for a "Project Manager".

Job Details:

- Project Manager**: Manages Projects
- VIEW**

Responsibility

Graduation Level

- Lorem ipsum, dolor sit amet consectetur adipisicing elit.
- Ipsam mollitia necessitatibus ducimus nisi ipsam possimus quas impedit eius accusantium nobis minus perspiciatis
- dolor sit amet consectetur adipisicing elit.
- nesciunt rem et vitae. Distinctio commodi maiores possimus.
- Ipsam mollitia necessitatibus ducimus nisi ipsam possimus quas

Skill Required

C# Java Manual Testing Automation Testing Manual Testing C# Java
Manual Testing Automation Testing Manual Testing

Pay: \$40000 **Deadline: 2022-09-08**

FIGURE 7: HR VIEW OF JOB PAGE



Jobs

Project Manager
Manages Projects
[VIEW](#)

Data Scientist
Manages Data Science Projects
[VIEW](#)

Data Analyst
Design Project
[VIEW](#)

Business Analyst
Deals with Business Projects
[VIEW](#)

Project Manager
Manages Projects
[APPLY NOW](#)

Responsibility
Graduation Level

- Lorem ipsum, dolor sit amet consectetur adipisicing elit.
- Ipsum mollitia necessitatibus ducimus nisi ipsam possimus quas impedit eius accusantium nobis minus perspiciatis
- dolor sit amet consectetur adipisicing elit.
- nesciunt rem et vitae. Distinctio commodi maiores possimus.
- Ipsum mollitia necessitatibus ducimus nisi ipsam possimus quas

Skill Required
C# Java Manual Testing Automation Testing Manual Testing C# Java Manual Testing
Automation Testing Manual Testing

Pay: \$40000 **Deadline:** 2022-09-08

FIGURE 8:EMPLOYEE VIEW OF JOB PAGE

Jobs [+ CREATE JOB](#)

Project Manager
Manages Projects
[VIEW](#)

Data Scientist
Manages Data Science Projects
[VIEW](#)

Data Analyst
Design Project
[VIEW](#)

Job Details

Position Name

Details

Responsibilities

Skill Required
C# Java Manual Testing Automation Testing Manual Testing [+ ADD](#)

Pay per month

Deadline
 yyyy - mm - dd

[CANCEL](#) [CREATE JOB](#)

FIGURE 9: CREATE NEW INTERNAL JOB CARD

Employees List

Employee Id	Employee name	Email	Dept	Position	Details
a@a.com	IT	Jr. Software Dev	View		
	IT	Software Dev	View		
	IT	Jr. Software Dev	View		
	IT	Jr. Software Dev	View		

Find column

- Employee Id
- Employee name
- Email
- Dept
- Position
- Details

[HIDE ALL](#) [SHOW ALL](#)

1–4 of 4

FIGURE 10: HR VIEW OF EMPLOYEES APPLIED FOR THIS JOB

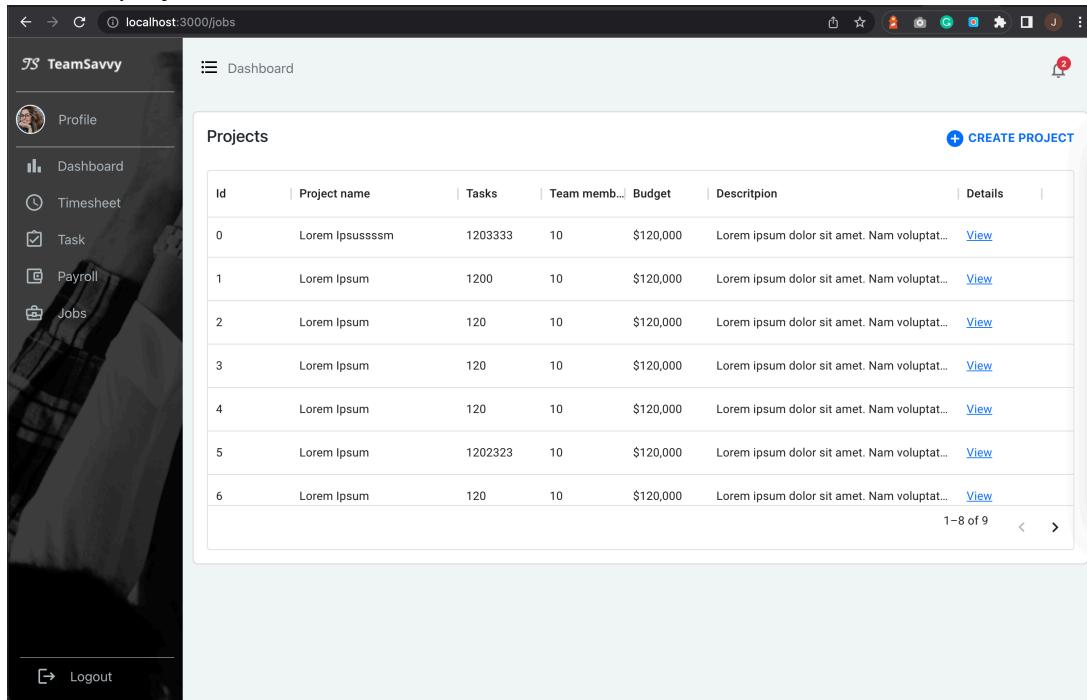


Front-end

We started working with react js, bootstrap for developing our front-end with responsive screens, we decided to start with role 'employee'. Pages we have achieved so far:

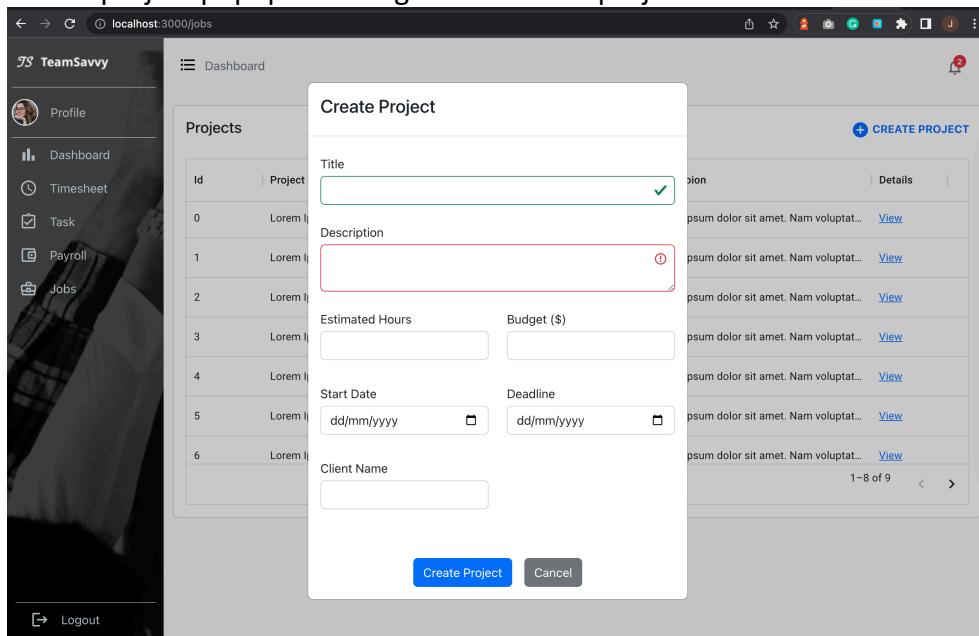
Project List:

This list depicts the information about the projects under manager or for HR total number of projects in organization. The screen will have project names, number of tasks assigned, number of team members in each project, project budget, project description and view link to show project details.



A screenshot of a web browser showing the 'TeamSavvy' application's dashboard. The left sidebar contains links for Profile, Dashboard, Timesheet, Task, Payroll, and Jobs. The main content area is titled 'Dashboard' and shows a table of 'Projects'. The table has columns for Id, Project name, Tasks, Team memb..., Budget, Description, and Details. Each row contains a 'View' link. A 'CREATE PROJECT' button is located at the top right of the table. The URL in the address bar is 'localhost:3000/jobs'.

Create project popup for manager to add new project in the list



A screenshot of a web browser showing the 'TeamSavvy' application's dashboard with a 'Create Project' modal open. The left sidebar is visible with its navigation links. The main content area shows a table of 'Projects' and the 'Create Project' modal. The modal has fields for Title (filled with 'New Project'), Description (empty), Estimated Hours, Budget (\$), Start Date, Deadline, and Client Name. There are 'Create Project' and 'Cancel' buttons at the bottom. The URL in the address bar is 'localhost:3000/jobs'.



Project details page is for managers to assign tasks to employees in his team. It also has a simple statistic to manage the process of project by showing how many tasks are still in progress and how many are completed.

The screenshot shows a web application interface for managing tasks. On the left is a sidebar with navigation links: Profile, Dashboard, Timesheet, Task, Payroll, and Jobs. Below the sidebar is a "Logout" button. The main area is titled "Project Name". At the top, there are three colored boxes showing task counts: "Tasks" (150), "In-Progress" (105), and "Completed" (40). Below these are three columns of task cards. Each card includes a title, a brief description, the assignee (John Doe), and a "MODIFY TASK" link. The "Completed" column also lists the completion date (Hours 12).

Create task for creating new task in current project and assign an employee.

The screenshot shows a "Create Task" modal window overlaid on the main dashboard. The modal has fields for "Title" (filled with "Create model for Database"), "Description" (empty), "Estimated Hours" (05), "Assign to" (Employee 1), and "Start Date" (yyyy-mm-dd). At the bottom are "Create Task" and "Cancel" buttons. The background dashboard shows the same task statistics and a list of tasks as the previous screenshot, with the "Completed" column showing the completion date (Hours 12).



Team Members:

The screen will show the team members in the project selected from the list and Bench and Add new employee will only be shown to HR. It will have employee name, Dept , position, salary, progress and manager can delete the team member from this table and view button to show details about each team member.

A screenshot of a web browser displaying the 'Team Members' page of the TeamSavvy application. The URL is localhost:3000/jobs. The left sidebar shows navigation options: Profile, Dashboard, Timesheet, Task, Payroll, and Jobs. The main content area has a header 'Team Members' with a 'ADD NEW EMPLOYEE' button. A dropdown menu 'Select Project' is open, showing 'Project 1'. Below is a table with four rows of data. The first three rows have a 'Dept' column value of 'IT', while the fourth row has a 'Dept' column value of 'Bench'. Each row includes columns for Id, Employee name, Status, Position, Salary, Progress (represented by a bar chart), Details (a blue 'View' link), and Delete (a red trash bin icon). At the bottom right of the table area, there is a pagination indicator '1-4 of 4'.

From view link the manager or HR can see employee's timesheet, task and profile.

A screenshot of a web browser displaying the 'Employee Name' page of the TeamSavvy application. The URL is localhost:3000/jobs. The left sidebar shows navigation options: Profile, Dashboard, Timesheet, Task, Payroll, and Jobs. The main content area has a header 'Employee Name' with tabs for 'Timesheet' (selected), 'Task', and 'Profile'. Below is a calendar for November 2022. Specific days are highlighted with colored boxes: November 1st is red ('Sick Leave(s)'), November 6th is orange ('Vacation Lea...'), November 7th is dark blue ('Clock in'), November 13th is green ('Sick Leave(s)'), November 14th is teal ('Sick Leave(s)'), November 15th is light blue ('Vacation Lea...'), and November 16th is grey ('Clock in'). To the right of the calendar is a 'Hours' section showing 'Worked' hours for each day: 40, 40, 40, 40, 40, 40, 40. At the bottom left, there is a 'Logout' link.



HR can assign project to an employee from bench button

The screenshot shows a web-based application titled "TeamSavvy" at the URL "localhost:3000/jobs". The left sidebar includes links for Profile, Dashboard, Timesheet, Task, Payroll, and Jobs, with "Jobs" currently selected. The main dashboard displays a "Team Members" section with a table showing four rows of placeholder data (Id 0-3, Employee Name "Lorem Ipsum"). A modal window titled "Assign Project" is open over the table. In the modal, the "Employee name" field is populated with "Jenny doe". The "Project" dropdown is set to "Project 1". The "Project Description" field contains placeholder text: "Lorem ipsum dolor sit amet. Nam volutpatibus tempore et distinctio natus eum magni quae est accusamus aspernatur.". At the bottom of the modal are two buttons: "Assign" (blue) and "Cancel". The background table has "View" and "Delete" actions for each row. The bottom right corner of the screen shows a system tray with a red notification badge containing the number "2".



HR can add new employee by clicking on add new employee and can add below given details.

TeamSavvy

Dashboard

Profile

Dashboard

Timesheet

Task

Payroll

Jobs

Logout

Personal Details

First Name:

Last Name:

Email:

Hire date:

DOB:

Phone:

Apartment:

Country:

Province:

City:

Postal Code:

Office Details

Department:

Role:

Extension:

Country:

Province:

City:

Skill Set

+ ADD

Bank Details

Bank Name:

Account Number:

Bank Code:

Pay scale

Salary per month:

Submit Changes

Cancel



Previous Project Implementations:

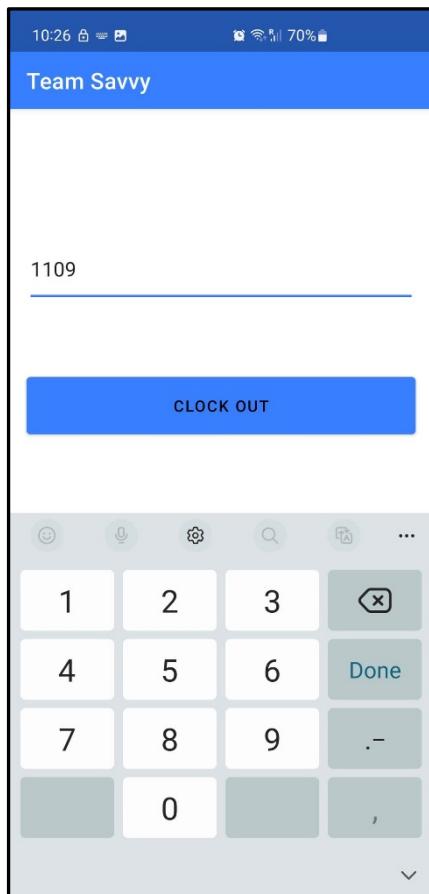
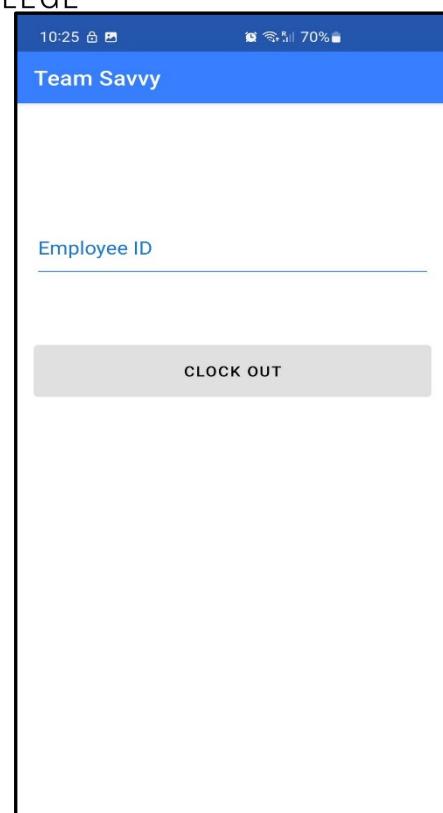
Android App for employee's biometric



Landing screen for an employee showcasing the project name, image, message for an employee for the attendance and loader.



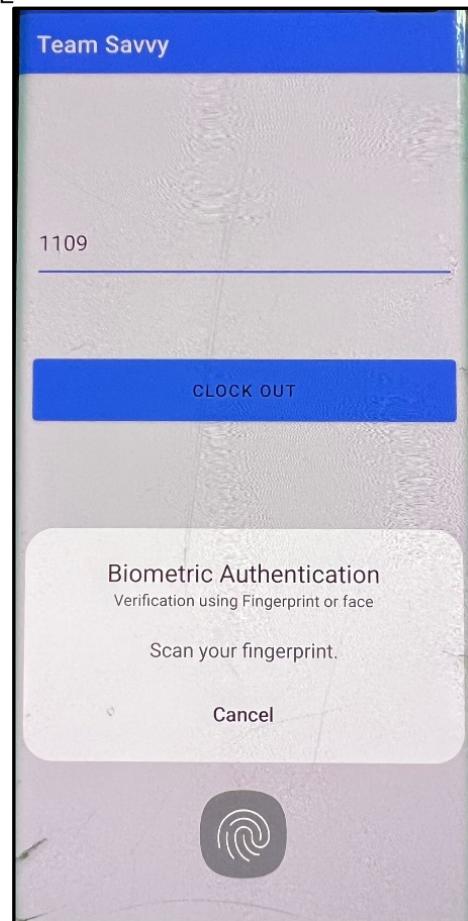
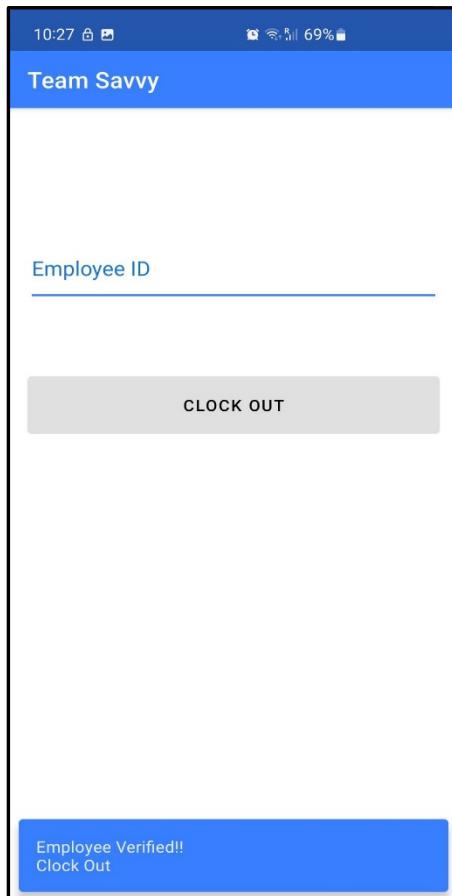
This screen will ask employee for their employee ID and allow him to clock-in or clock-out.
We are still working on design and few features.



After entering the employee id the button will be activated.



The app will ask for biometric verification, if the fingerprint matches to the record of fingerprint in device the app will accept and allow user to clock-in or clock-out.



At last, the app will show the snack bar to show if an employee has been verified or not also displays the status clock-in or clock-out.



APIs

We have created API in .NET CORE using Entity Framework, LINQ, Automapper and swagger documentation.

A screenshot of a web browser showing the Swagger UI for the Team Savvy Web API. The title bar says "Swagger UI". The address bar shows "localhost:44362/swagger/index.html". A dropdown menu "Select a definition" is set to "Team Savvy Web Api Version: 1.0". The main content area is titled "Team Savvy Web API v1 OAS3" and shows a list of API endpoints under categories like Auth, Dropdowns, Email, Employee, EmployeeTask, Leaves, and Payroll. An "Authorize" button is visible at the top right of the content area.

A screenshot of the same Swagger UI interface, but with a different set of API endpoints listed under categories such as Auth, Dropdowns, Email, Employee, EmployeeTask, Leaves, and Payroll. The "Authorize" button is again visible at the top right.

A screenshot of the Swagger UI interface, showing a different set of API endpoints under categories like Auth, Dropdowns, Email, Employee, EmployeeTask, Leaves, Payroll, Projects, and TimeSheet. At the bottom, there is a section labeled "Schemas". The "Authorize" button is visible at the top right.

This is the swagger documentation to give a user interface to an API so developer can test API like postman. In this interface we can test all the request like post, delete, update, and get request. Above are the endpoints for Auth (Authorization), Dropdown, Email, Employee, Employee Task, Leaves, Payroll, Projects and Timesheet controllers.



Swagger UI x +

localhost:44362/swagger/index.html

Student Bookmarks json Mobile Dev II ACX Associate Promise... Paraphrasing Tool J... Bell Bottom Visual Studio Code... Education - Micros... How to Integrate G... Other bookmarks

Auth

POST /api/Auth/login

Dropdowns

GET /api/Dropdowns/countryProvinceCity

Email

POST /api/Email

Employee

GET /api/Employee

GET /api/Employee/{employeeId}

POST /api/Employee/addEmployee

PUT /api/Employee/updateEmployee

DELETE /api/Employee/deleteEmployee/{id}/{status}

Swagger UI x +

localhost:44362/swagger/index.html

Student Bookmarks json Mobile Dev II ACX Associate Promise... Paraphrasing Tool J... Bell Bottom Visual Studio Code... Education - Micros... How to Integrate G... Other bookmarks

EmployeeTask

GET /api/EmployeeTask

GET /api/EmployeeTask/employeeId/{employeeId}

GET /api/EmployeeTask/{taskName}

GET /api/EmployeeTask/taskId/{taskId}

POST /api/EmployeeTask/addTask

POST /api/EmployeeTask/addTasks

PUT /api/EmployeeTask/updateTask

DELETE /api/EmployeeTask/deleteTask/{taskId}

Leaves

GET /api/Leaves/leaveId/{leaveId}

GET /api/Leaves/employeeId/{employeeId}

POST /api/Leaves/addLeave



Swagger UI localhost:44362/swagger/index.html

Student Bookmarks json Mobile Dev II ACX Associate Promise... Paraphrasing Tool J... Bell Bottom Visual Studio Code... Education - Micros... How to Integrate G... Other bookmarks

Payroll

GET	/api/Payroll	▼ 🔒
GET	/api/Payroll/list/employeeId/{employeeId}	▼ 🔒
GET	/api/Payroll/payrollId/{payrollId}	▼ 🔒
GET	/api/Payroll/employeeId/{employeeId}	▼ 🔒
POST	/api/Payroll/addPayroll	▼ 🔒
PUT	/api/Payroll/updatePayroll	▼ 🔒
DELETE	/api/Payroll/deletePayroll	▼ 🔒

Projects

GET	/api/Projects	▼ 🔒
GET	/api/Projects/employeeId/{employeeId}	▼ 🔒
GET	/api/Projects/{projectName}	▼ 🔒
GET	/api/Projects/projectId/{projectId}	▼ 🔒

Swagger UI localhost:44362/swagger/index.html

Student Bookmarks json Mobile Dev II ACX Associate Promise... Paraphrasing Tool J... Bell Bottom Visual Studio Code... Education - Micros... How to Integrate G... Other bookmarks

Payroll

DELETE	/api/Payroll/deletePayroll	▼ 🔒
--------	----------------------------	-----

Projects

GET	/api/Projects	▼ 🔒
GET	/api/Projects/employeeId/{employeeId}	▼ 🔒
GET	/api/Projects/{projectName}	▼ 🔒
GET	/api/Projects/projectId/{projectId}	▼ 🔒
POST	/api/Projects/addProject	▼ 🔒
PUT	/api/Projects/updateProject	▼ 🔒
DELETE	/api/Projects/deleteProject/{projectId}	▼ 🔒

TimeSheet

GET	/api/TimeSheet/{employeeId}	▼ 🔒
GET	/api/TimeSheet/mobile/{employeeId}	▼ 🔒
POST	/api/TimeSheet/addTimesheet	▼ 🔒

Above screenshots showing separate endpoints (GET,POST,PUT,DELETE) of all controllers created.



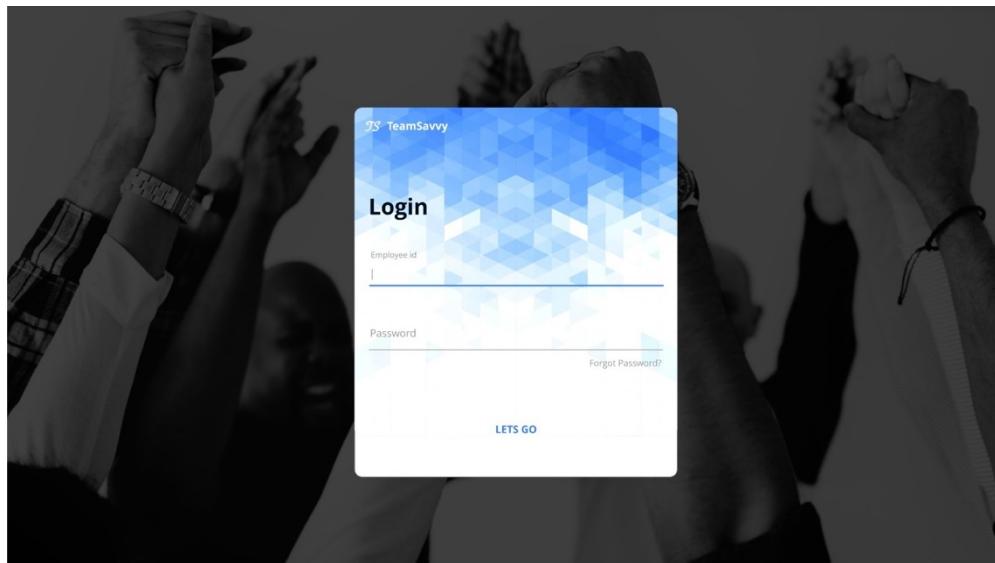
UI Design using Adobe XD

The snapshots below shows the flow of our application. Design Link:

<https://xd.adobe.com/view/732c0d35-1e41-41d7-876c-25bde1a844a9-267f/?fullscreen>

Login:

The employee and admin will login with the predefined credentials provided by organization.



Timesheet:

This screen will show the information of employee clock-in and clock-out, with total hours they worked per week. Additionally, employee can apply for leaves and it will be displayed on the calendar

A screenshot of the TeamSavvy Timesheet interface. On the left is a sidebar with navigation links: "Emp name", "Timesheet" (selected), "Task", "Payroll", "Internal Jobs", and "Dashboard". The main area shows a calendar for October 2022, with days from 25 to 31. Each day has "IN" and "OUT" times listed. Some days have colored boxes indicating leave types: orange for "Sick Leave" and yellow for "Casual Leave". To the right of the calendar is a vertical "Hours" column showing totals: 40 for the first week and 30.5 for the second week. The top right of the interface includes a search icon and a red notification dot.



Task:

Task screen will have the assigned task for an employee, which he can modify to add hours, start date, and end date. Employee can also change the status from assigned to in-progress and then by using update status in in-progress lane he can change status to completed.

A screenshot of the TeamSavvy application interface showing the Task screen. The interface is divided into three main columns: "Assigned task", "In Progress", and "Completed".

- Assigned task:** Contains three "Create model for Database" sections and three "MODIFY TASK" buttons.
- In Progress:** Shows one entry: "Hours 12" with an "UPDATE STATUS" button.
- Completed:** Shows one entry: "Hours 12".

The sidebar on the left includes links for Timesheet, Task, Payroll, Internal Jobs, and Dashboard.

Payroll:

This screen will have the collection of pay slips with information of each month, total hours, and earnings, from here user can view each pay slip. The Pay Slip contains all the Earnings of an employee at the company from the date he has joined.

A screenshot of the TeamSavvy application interface showing the Payroll screen. It displays a table of "Payslips" with the following data:

Pay Date	Pay type	Total Hours	Earnings	Net Pay(after taxes)	File
31-Jan-2022	Monthly	80	\$3600	\$3200	VIEW FILE
28-Feb-2022	Monthly	80	\$3100	\$2800	VIEW FILE
31-Jan-2022	Monthly	80	\$3600	\$3200	VIEW FILE
28-Feb-2022	Monthly	80	\$3100	\$2800	VIEW FILE
31-Jan-2022	Monthly	80	\$3600	\$3200	VIEW FILE
28-Feb-2022	Monthly	80	\$3100	\$2800	VIEW FILE
31-Jan-2022	Monthly	80	\$3600	\$3200	VIEW FILE
28-Feb-2022	Monthly	80	\$3100	\$2800	VIEW FILE

Rows per page: 20 ▾ 1-8 of 8

Internal Jobs:

This screen will display employee with the internal jobs postings which he can read the description and apply for the jobs.



The image shows the TeamSavvy mobile application interface. The top navigation bar includes the 'TeamSavvy' logo, a search icon, and a notification bell icon. On the left, a vertical navigation menu lists 'Emp name', 'Timesheet', 'Task', 'Payroll', 'Internal Jobs', and 'Dashboard'. The main content area displays four job listings under the heading 'Jobs': 'Assistant Manager' (with a note about Employee, Leaves, Payroll), 'VIEW' button; 'Assistant Manager' (with a note about Employee, Leaves, Payroll), 'VIEW' button; 'Assistant Manager' (with a note about Employee, Leaves, Payroll), 'VIEW' button; and 'Assistant Manager' (with a note about Employee, Leaves, Payroll), 'VIEW' button. Each listing includes a 'Skills required' section with orange circular buttons for 'C#' and 'Java', and 'Automation Testing' and 'Manual Testing' sections. At the bottom, it shows a 'Pay' of '40-50/ hour' and a 'Deadline' of '25 Oct, 2022'.

User Profile:

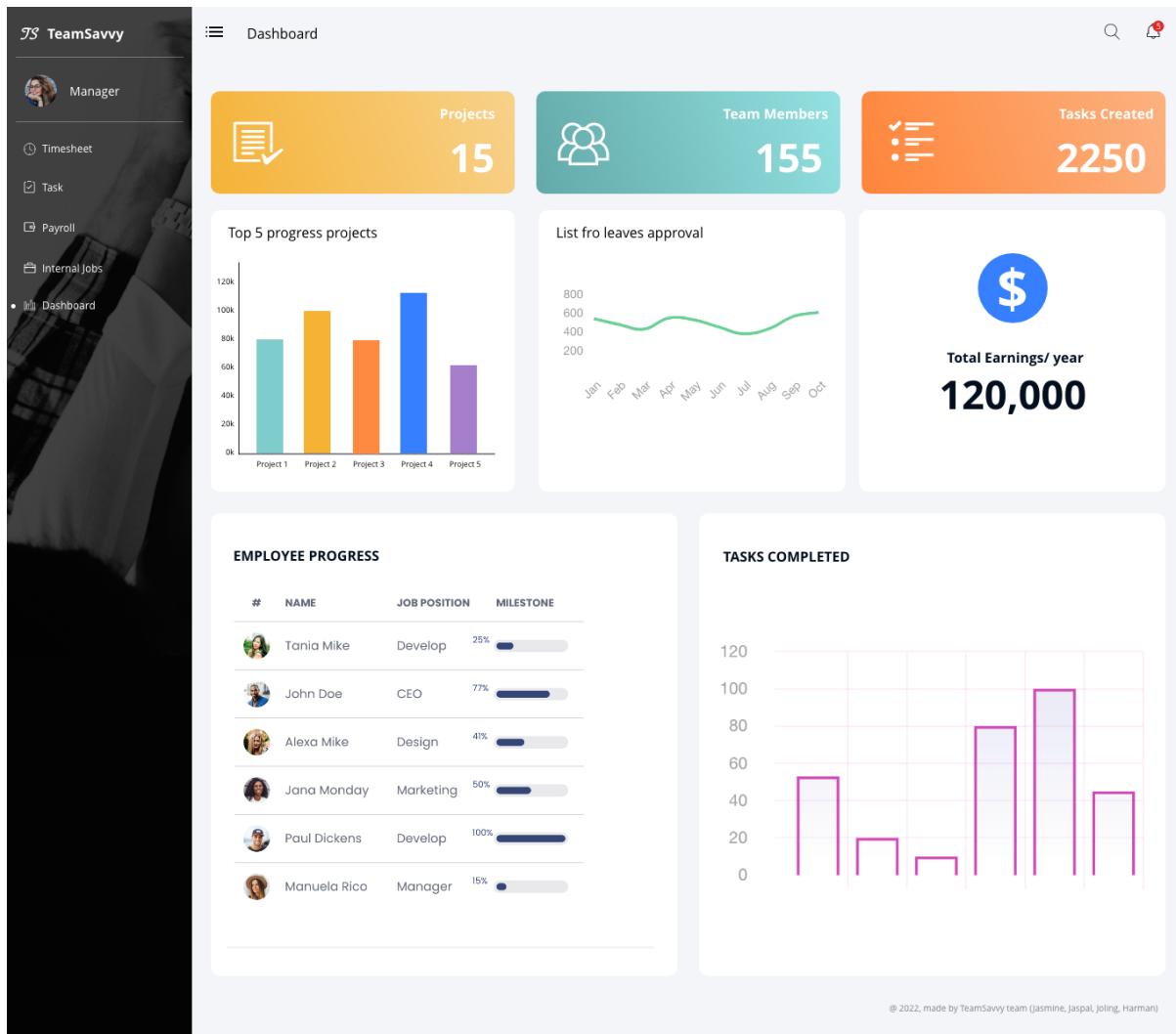
This screen will show the personal details, team members, position, department, skill set and bank details of an employee.

Dashboard:



DOUGLAS COLLEGE

This screen is for manager and HR, here they can view projects, team members, tasks, and other information. This screen will also have feature to create graphs widgets with predefined dropdowns. Manager can download, modify and delete the graph.



Front-end

We started working with react js, bootstrap for developing our front-end with responsive screens, we decided to start with role 'employee'. Pages we have achieved so far:

Login:

This screen will allow users to login with credentials provided by organization or can reset password using OTP



localhost:3001

A black and white photograph of several people's hands raised and clapping, serving as the background for the login page.

Login

Employee Id

Password

[Forgot Password?](#)

LETS GO

localhost:3001/forgotPassword

A black and white photograph of several people's hands raised and clapping, serving as the background for the forgot password page.

Forgot Password

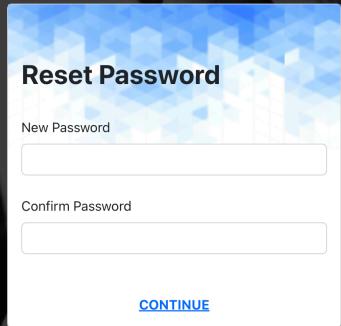
Enter OTP

OTP sent on your email

CONTINUE



localhost:3001/resetpassword

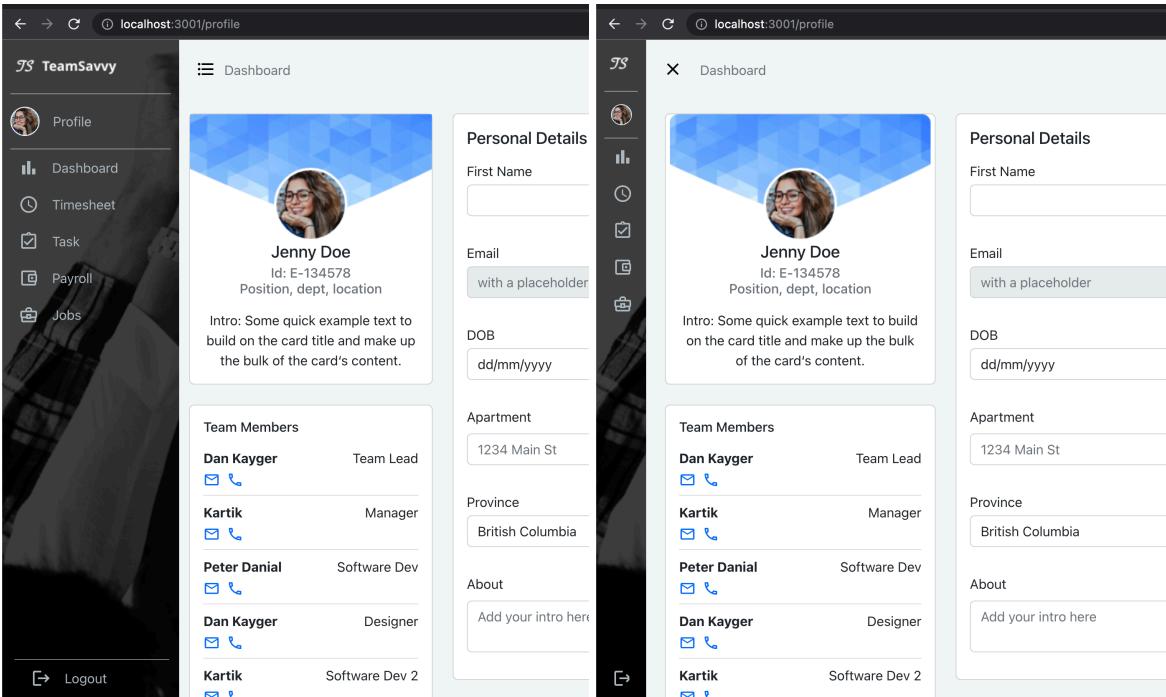


A modal window titled "Reset Password" is centered over a background image of several people's hands raised in a gesture of support or participation. The modal contains fields for "New Password" and "Confirm Password", both with placeholder text, and a "CONTINUE" button at the bottom.

Sidenav:

The animated sidebar with navigation links to access the platform.

localhost:3001/profile



The image shows two side-by-side browser windows illustrating the Sidenav feature. On the left, the "Open" state is shown where a dark sidebar is pulled out from the left, containing a "Profile" section with icons for Dashboard, Timesheet, Task, Payroll, and Jobs, and a "Logout" link at the bottom. The main content area displays a "Dashboard" card for "Jenny Doe" and a "Team Members" section listing Dan Kayger, Kartik, Peter Danial, Dan Kayger, and Kartik. On the right, the "Closed" state is shown where the sidebar is collapsed into a thin vertical bar on the far left. The main content area remains the same, showing the "Dashboard" card and the "Team Members" list.

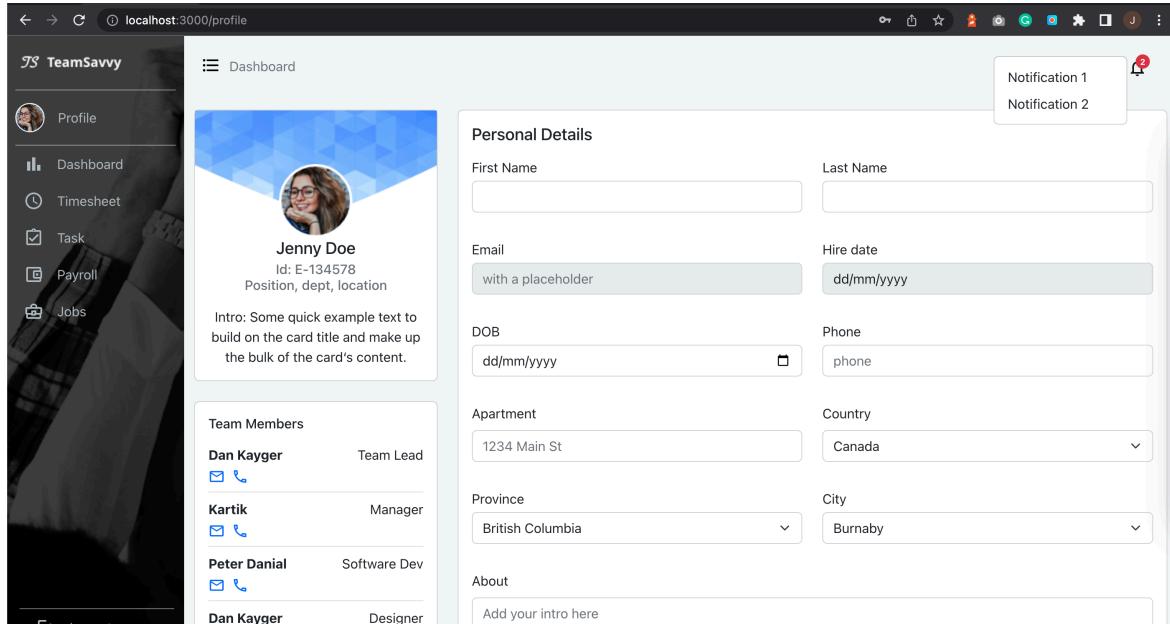
Open

Closed



Header and Navigation:

Header has dashboard title which will be breadcrumb for navigation and notification with indicator, it is hardcoded for now. We will work on it after midterm.



A screenshot of a web application interface titled "localhost:3000/profile". The left sidebar contains a "Profile" section with a user icon and the name "TeamSavvy". Below it is a navigation menu with links: Dashboard, Timesheet, Task, Payroll, and Jobs. The main content area is titled "Dashboard". It features a large card for "Jenny Doe" with a profile picture, the name "Jenny Doe", ID "E-134578", and placeholder text "Position, dept, location". Below this is a "Team Members" section listing four team members: Dan Kayger (Team Lead), Kartik (Manager), Peter Danial (Software Dev), and Dan Kayger (Designer). To the right is a "Personal Details" form with fields for First Name, Last Name, Email, Hire date, DOB, Phone, Apartment, Country, Province, City, and About. A red notification bubble in the top right corner indicates "Notification 1" and "Notification 2".



User Profile:

The user profile page has employee information, which consists of Personal details, Team members with their contact details, Skill set and add new skill option, bank details, and resign option. From here HR or manager can get the overview of an employee and what skills they acquire.

TS TeamSavvy

- Profile
- Dashboard
- Timesheet
- Task
- Payroll
- Jobs

[Logout](#)

Dashboard



Jenny Doe
Id: E-134578
Position, dept, location

Intro: Some quick example text to build on the card title and make up the bulk of the card's content.

Team Members

Name	Role
Dan Kayger	Team Lead
Kartik	Manager
Peter Danial	Software Dev
Dan Kayger	Designer
Kartik	Software Dev 2
Dan	Tester

Personal Details

First Name	Last Name
<input type="text"/>	<input type="text"/>
Email	Hire date
<input type="text"/> with a placeholder	<input type="text"/> dd/mm/yyyy
DOB	Phone
<input type="text"/> dd/mm/yyyy	<input type="text"/> phone
Apartment	Country
<input type="text"/> 1234 Main St	<input type="text"/> Canada
Province	City
<input type="text"/> British Columbia	<input type="text"/> Burnaby
About	
<input type="text"/> Add your intro here	

Skill Set

C# Java Manual Testing Automation Testing Manual Testing C# Java

Manual Testing Automation Testing Manual Testing

[+ ADD](#)

Bank Details

Bank Name	Account Number
<input type="text"/>	<input type="text"/>
Bank Code	
<input type="text"/>	

Resignation

[Resign](#)

[Submit Changes](#) [Cancel](#)



Timesheet:

The Calendar is generated with react-big-calendar which has custom tags for leaves and popups for deleting existing leaves and add new leave. Also have clock-in and clock-out indicator. The page also has Worked total hours per week for an employee.

A screenshot of a web-based timesheet application. On the left is a sidebar with a dark background and white icons for Profile, Dashboard, Timesheet, Task, Payroll, and Jobs. Below the sidebar is a "Logout" button. The main area is titled "Dashboard". It features a "back" and "next" button above a "October 2022" calendar grid. The grid shows dates from Sunday, October 25, to Saturday, October 31. A red box highlights the number "01" in the Saturday cell, with a tooltip "Sick Leave(s)". To the right of the calendar is a vertical sidebar titled "Hours" with a "Worked" section showing the value "40" in blue. There are three horizontal bars below the "Worked" section, each also showing the value "40" in blue.

Task:

The task page has feature to show assigned, in progress and completed tasks for an employee. We are assuming the employee only working in one project at a time. On click of modify task user will get pop-up to add hours and change status of the task. On update status, the employee can only change status to completed task.

A screenshot of a web-based task management application. The sidebar on the left is identical to the timesheet sidebar. The main area is titled "Dashboard". It displays three columns: "Assigned Task", "In progress", and "Completed". Each column contains a list of tasks, each with a "MODIFY TASK" link. The "Assigned Task" column lists three tasks, all of which are "Create model for database". The "In progress" column lists two tasks, both of which are "Create model for database" and have a "Hours 12" entry. The "Completed" column lists one task, "Create model for database", which also has a "Hours 12" entry. Each task row also has a "UPDATE STATUS" link.



Payroll:

The payroll has payslips screens where employee can check the list of paystubs and also can view the paystubs individually by clicking on view file.

The screenshot shows two views of the TeamSavvy payroll system. The top view displays a list of paystubs (Paylips) with columns for Pay Date, Pay Type, Total Hours, Earnings, Net Pay(after taxes), and a 'VIEW FILE' link. The bottom view is a detailed paystub for Joling Weng, showing personal information (Employee#300335548, Department: IT, Start Date 1-Jan-2022, End Date), address (12546, 56 Ave Surrey, BC, H7Y76), organization (Organization name 1868, XYZ road, Ontario, ON, D54F54), earnings (Reg Hours, Sick, Vac Pay), net (Summary, Current, Year to Date), and pay distribution (Type, Amount, Account#, Bank).

Paylips

Pay Date	Pay Type	Total Hours	Earnings	Net Pay(after taxes)	File
31-MAR-2022	Monthly	80	\$3600	\$3200	VIEW FILE
28-FEB-2022	Monthly	80	\$3600	\$3200	VIEW FILE
31-JAN-2022	Monthly	80	\$3600	\$3200	VIEW FILE
31-JAN-2022	Monthly	80	\$3600	\$3200	VIEW FILE
31-JAN-2022	Monthly	80	\$3600	\$3200	VIEW FILE
31-JAN-2022	Monthly	80	\$3600	\$3200	VIEW FILE
31-JAN-2022	Monthly	80	\$3600	\$3200	VIEW FILE
31-JAN-2022	Monthly	80	\$3600	\$3200	VIEW FILE

Joling Weng Employee#300335548 Department: IT Start Date 1-Jan-2022 End Date

Address: 12546, 56 Ave Surrey, BC, H7Y76 Organization: Organization name 1868, XYZ road, Ontario, ON, D54F54

EARNINGS

Type	Hours	Rate	Amount	YTD
Reg Hours	80	23	1840	3800
Sick	6.0	20	180	350
Vac Pay	0.00	30.00	30.00	30.00

NET

Summary	Gross Pay	Deductions	Net Pay
Current	3800	500	3200
Year to Date	20000	3500	16500

PAY DISTRIBUTION

Type	Amount	Account#	Bank
Electronic bank transfers	\$3100	1240-1102-1111	CIBC
Electronic bank transfer	\$3000	1240-1102-1111	RBC



Jobs:

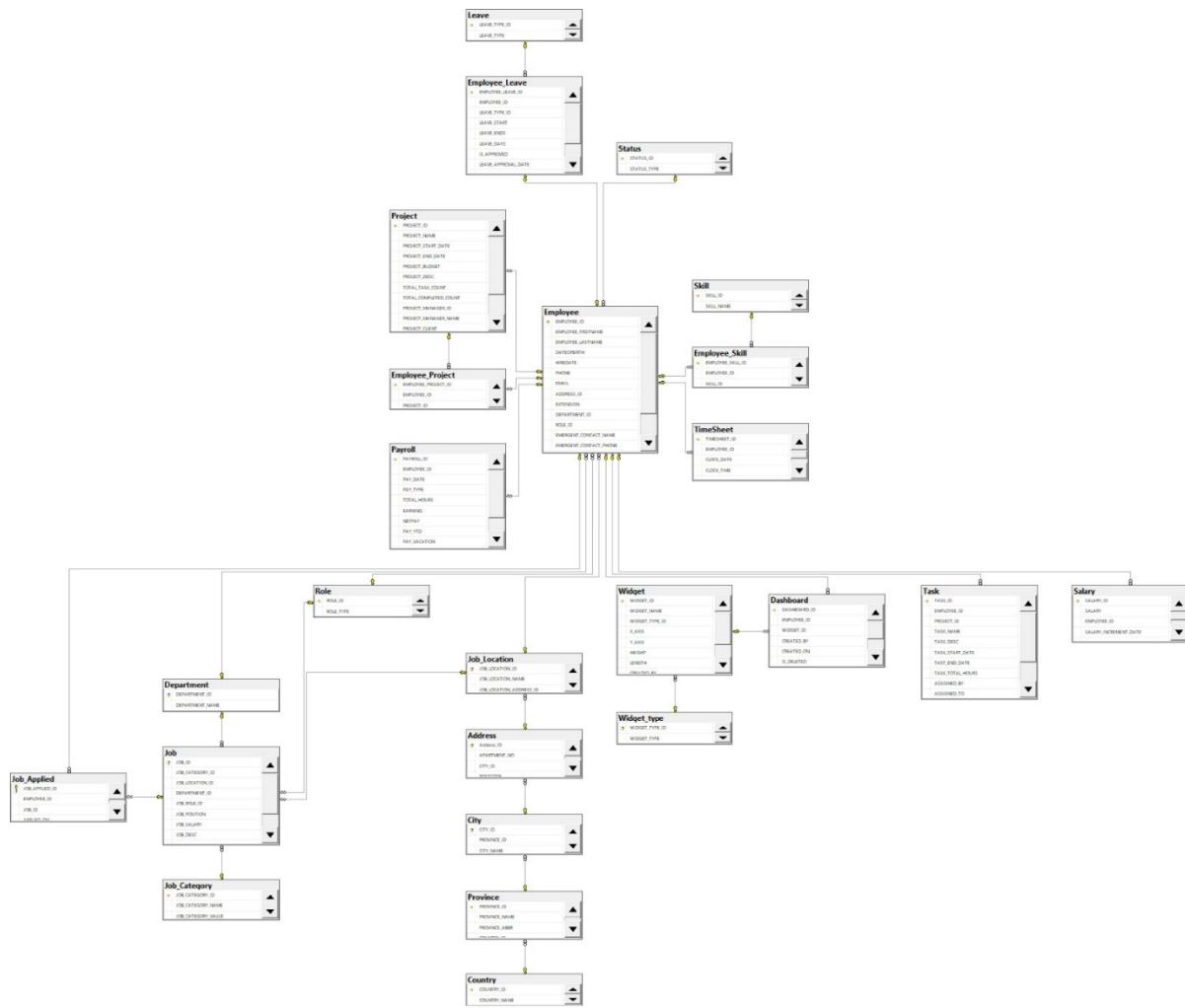
The jobs page for employee have internal jobs created by HR and employee will not see the create job option. Employee can directly apply using Apply now button. HR will get the notification on their portal from where they can check the profile of an employee who applied.

The screenshot shows a web browser window for 'localhost:3000/jobs'. On the left, a sidebar menu for 'TeamSavvy' lists: Profile, Dashboard, Timesheet, Task, Payroll, and Jobs. The 'Jobs' item is selected. The main content area has a header 'Dashboard' and a 'Jobs' section with a '+ CREATE JOB' button. Below this, three job listings are shown: 'Assistant Manager Dept-A', 'Assistant Manager Dept-B', and 'Assistant Manager Dept-C'. Each listing includes a 'VIEW' link. To the right, a detailed view of the 'Assistant Manager' job is displayed. It features a 'CREATE JOB' button, a description of the role, a 'Responsibility' section with a bulleted list, a 'Skill Required' section with several skill tags (C#, Java, Manual Testing, Automation Testing), and a 'Pay: \$3500' and 'Deadline: 25 Oct, 2022' at the bottom.

Database

We've created all the Tables that are required for the functioning of the Application. These tables interlinked with each other has all the attributes that gives us the glimpse of how the Application will work.

Structure of the Database:

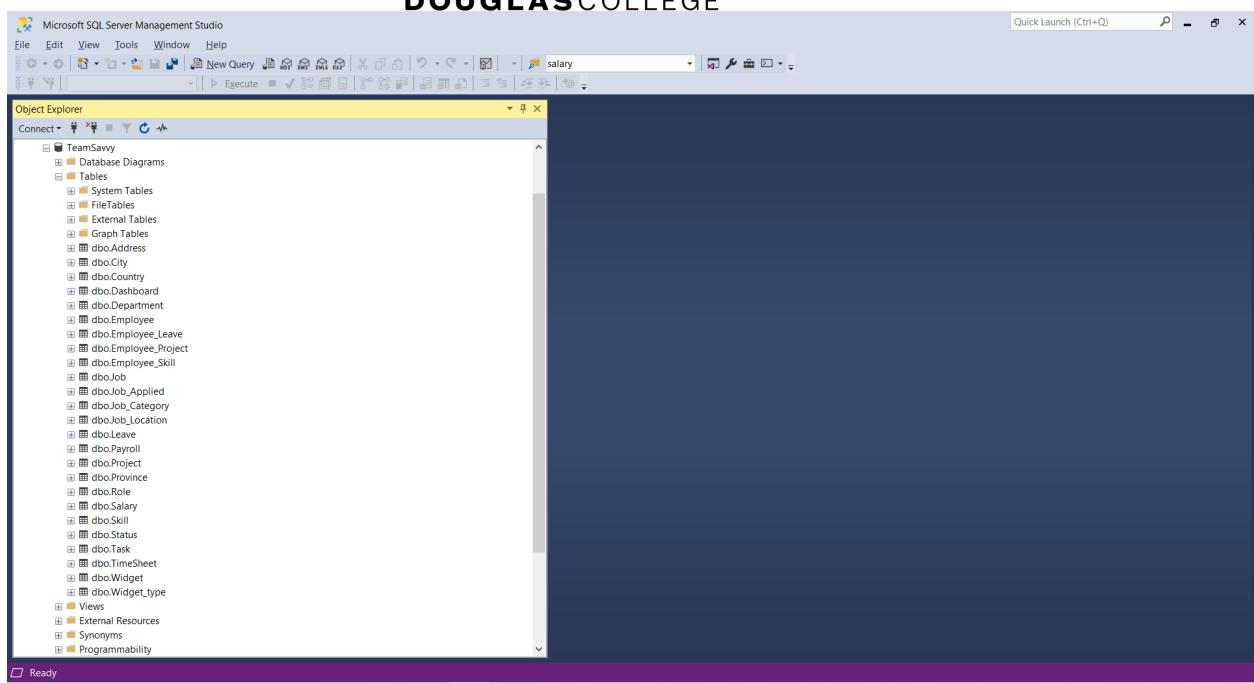


Tables in our database:

- **Employee**: PK-EmpID, FK – AddressID, DepartmentID, ProjectID, RoleID, StatusesID, First Name, Last Name, DOB, HireDate, Phone, Extension, Email, Bank Account No.
- **Project**: PK- ProjectID, FK – Project Manager ID, Project Name, Project Start Date, Project End date, Project Budget, Project Description, Total Task Count, Total Inprogress Count, Total Completed Count, Project Manager, Project Client, Project Load, Project Total Employee
- **Leave**: PK – Leave Type ID, FK – Employee ID, Leave Start Date, Leave End Date, Leave Days, Leave Approval Date, Leave Approved By, Leave Status
- **Employee_Leave**: PK – Employee Leave ID, FK – Employee Type ID



- **Employee_Project:** PK – Employee Project ID, FK – Employee ID, Project ID
- **Payroll:** PK – Payroll ID, FK – Employee ID, Pay ID, Pay Date, Pay Type, Total Hours, Earning, Net Pay, Pay_Vacation, Pay_Sick
- **Status:** PK – Status ID, Status_Type
- **Skill:** PK – Skill ID, Skill Name
- **Employee_Skill:** PK – Employee Skill ID, FK – EmployeeID, SkillID
- **Time_Sheet:** PK – TimesheetID, FK – EmployeeID
- **Role:** PK – RoleID, RoleName
- **Department:** PK – DepartmentID, DepartmentName
- **Job:** PK – JobID, FK – JobCategoryID , JobLocationID, DepartmentID, JobRoleID, JobSalary, JobDesc
- **Job Applied:** PK- Job Applied ID, FK – EmployeeID, JobID
- **Job Category:** PK – JobCategoryID, JobCategoryName, JobCategoryValue
- **Job Location:** PK – JobLocationID, FK – CityID, Location, PostalCode
- **Address:** PK – AddressID, FK - CityID, Apartment, Postal Code
- **City:** PK – CityID, CityName, FK – Province ID
- **Province:** PK – Province ID, FK – Country ID, ProvinceName, ProvinceAbbreviations
- **Country:** PK – CountryID, Country Name
- **Dashboard:** PK – DashboardID, FK – EmployeeID, WidgetID
- **Widget:** PK – WidgetID, FK – WidgetTypeID, WidgetName)
- **Widget_Type:** PK – WidgetTypeID, WidgetType
- **Task:** PK – TaskID, FK – EmployeeID, FK-ProjectID, TaskName, TaskDescription, TaskStartDate, TaskEndDate, TaskTotalHours, AssignedBy, AssignedTo, AssignedDate, TaskStatus
- **Salary:** PK – (SalaryID, SalaryIncdate), FK – EmployeeID, SalaryType, Salary



Above screenshot is about TeamSavvy database with all the tables required for application as of now and attaching the SQL script to generate the above database with tables and populate data.

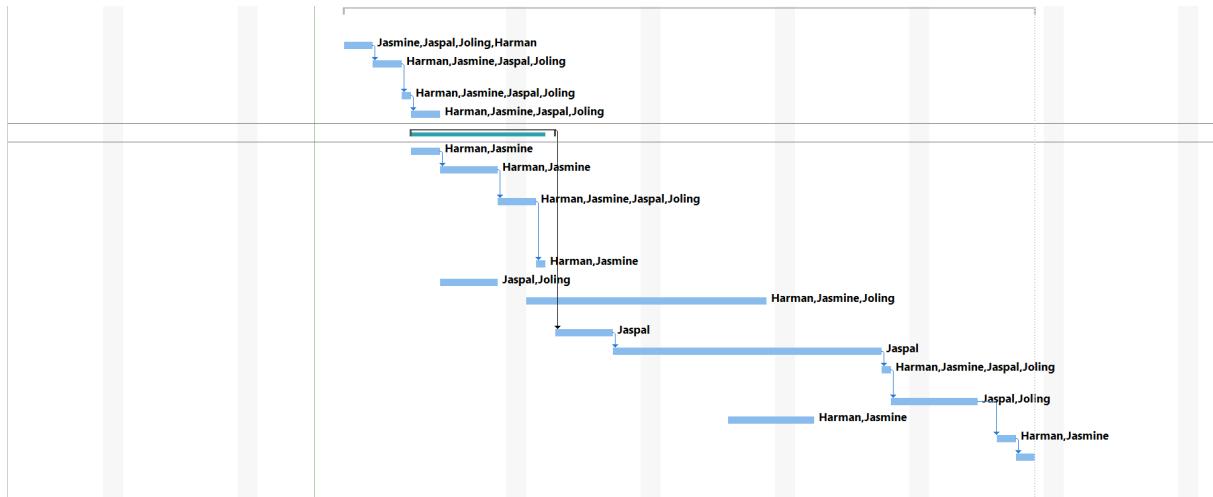
Pending Implementations:

A Gantt chart, a Network Diagram and Timeline on Microsoft Project:

Network Diagram: We have completed the content under the green box so far. Every other task is in progress.

Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names	Add New Column
Software Development	Brainstorming	3 days	Wed 9/21/22	Fri 9/23/22		Jasmine,Jaspal,Joling,Harman	
	Analysis/Software Requirements	3 days	Sat 9/24/22	Mon 9/26/22	1	Harman,Jasmine,Jaspal,Joling	
	Scope Determination	1 day	Tue 9/27/22	Tue 9/27/22	2	Harman,Jasmine,Jaspal,Joling	
	Feasibility Analysis	3 days	Wed 9/28/22	Fri 9/30/22	3	Harman,Jasmine,Jaspal,Joling	
Design	Interface Analysis	3 days	Wed 9/28/22	Fri 9/30/22		Harman,Jasmine	
	User Interface design	5 days	Sat 10/1/22	Thu 10/6/22	6	Harman,Jasmine	
	Incorporate feedback into functional specifications	2 days	Fri 10/7/22	Mon 10/10/22	7	Harman,Jasmine,Jaspal,Joling	
	Design complete	1 day	Tue 10/11/22	Tue 10/11/22	8	Harman,Jasmine	
	Database Design	5 days	Sat 10/1/22	Thu 10/6/22		Jaspal,Joling	
	Front-end Development	20 days	Mon 10/10/22	Thu 11/3/22		Harman,Jasmine,Joling	
	Database Scripting	5 days	Thu 10/13/22	Tue 10/18/22	5	Jaspal	
	API Development	20 days	Wed 10/19/22	Tue 11/15/22	12	Jaspal	
	UI and API integration	1 day	Wed 11/16/22	Wed 11/16/22	13	Harman,Jasmine,Jaspal,Joling	
	Integration Testing	7 days	Thu 11/17/22	Fri 11/25/22	14	Jaspal,Joling	
	Functional testing	7 days	Mon 10/31/22	Tue 11/8/22		Harman,Jasmine	
	Documentation	2 days	Mon 11/28/22	Tue 11/29/22	15	Harman,Jasmine	
	Deployment	2 days?	Wed 11/30/22	Thu 12/1/22	17		

Gantt chart:



Timeline: We have completed the content under the green box so far.



Dashboard, Timesheet UI, and logic

- The dashboard requires to use additional chart tools and the data fetching and queries are more complicated. Hence, we need more time on this part.
- Timesheet page shows employee leaves, clock in/ clock out time and, allowing employee to manage leaves. We're still figuring out how to implement the logic and debugging.

Task, Job

- Logic for task page is almost done. There is a status update issue with the connection to API. We will fix the logic before next report.
- For job page, we still haven't implemented the connection for posting new jobs from HR and the logic when the employee clicks on apply now. Also, the applied button in HR view will show the corresponding employees' profile.

Database

- Since we were focusing on the UI and logic, we haven't populated enough data for showing and testing. We will insert more data to show so that the application will look more alive and reasonable.



- Also, while we're doing the logic and connection, we found out that we might need to change the database a little to make the application more user-friendly. We will be keeping optimizing it.

Proposed Revisions:

As shown in the previous report, due to the restriction of device, we will only use mobile app for clock in/ clock out purpose for each employee without saving the biometric information to user database. This is because Android has restricted it and only shares the function for using.