

[**Introduction**](#_xnl2flscf7hz) **4**

[**Requirement Analysis**](#_2q48tqt6dipe) **5**

[Gathering requirements](#_mxdh04al3miv) 5

[High Level Plan](#_h6277zx5j6fp) 6

[**Use Cases**](#_2vovzcwl1sgj) **7**

[**System Analysis & Design**](#_9tszw11wn9jm) **9**

[Database Design](#_k29zt7u4ym5v) 9

[Component Diagram](#_ks36vep47phn) 12

[Technology Stack](#_hza9d4n84348) 13

[Libraries used](#_bv2cq1w6cgzs) 13

[Sequence Diagram](#_livfmssaf59d) 15

[Screen Design](#_rfco4ukwv3ew) 16

[**Implementation**](#_9pqdwkhukhxq) **20**

[Login Page](#_9ke5qgfdfzio) 20

[Dashboard](#_ppglt3fhzkl2) 21

[Add Leave modal](#_hxou0y45tov3) 21

[Resources](#_3oh8f2sle1va) 23

[Risk Chart](#_529laxb3bbot) 25

[**Testing**](#_eqassnb60y4w) **26**

[Front End Application](#_h914wnkdn0j6) 26

[Test Case - UX01](#_bgbv4jrv3s0a) 26

[Test Case - UX02](#_glhf821rzuzf) 27

[Test Case - UX03](#_lus6asi1z0uf) 28

[Test Case - UX04](#_hxkks3awzpqb) 28

[Test case - UX05](#_nxdy7pg398tf) 29

[Back-End Application](#_tldvcom2f7pc) 30

[Test Case - BackEnd01](#_ph9dkzhqgqu0) 30

[Test Case - BackEnd02](#_8xqsews8wwtw) 30

[Test Case - BackEnd03](#_a4tpefl2inwl) 31

[Test Case - BackEnd04](#_hekoe1vmafnq) 31

[Test Case - BackEnd05](#_xtk7pcmb3yo3) 32

[**Limitations**](#_i3p5arwpetia) **33**

[**Future Enhancements**](#_p88v96ya7oko) **34**

[**Conclusion**](#_yq56dwhzh5xz) **35**

Figure 1 : Entity Relationship Diagram 8

Figure 2 : Component Diagram 11

Figure 3 : Sequence Diagram 14

Figure 4 : Wireframes 15

Figure 5 : Screenshots 19

Table 1 : High Level Plan 5

Table 2 : Use Cases 6

Table 3 : Entity Description 9

Table 4 : Server-side Libraries 12

Table 5 : Client-side Libraries 13

Table 6 : Test Case - Front-end 26

Table 7 : Test Case - Backend 30

# Introduction

Every organization needs to manage its projects, teams and resources (employees). It consists of many teams, which in turn consists of many employees (resources).

For each team, there is a specific number of resources that need to be available everyday, in order for the work to progress smoothly. In case the number of available resources drops below this minimum value required, it might cause problems for the team.

When applying for a leave, a resource needs to take these factors into consideration.

What is this system?

Riskly, a Leave Risk Tracker, facilitates the user to track leaves, manage resources and also view reports. Each team has a threshold value, which specifies the minimum number of available resources required on each day. If a specific leave causes available resources of a team to drop below the threshold, it creates a risk.

Why this system?

Current system has drawbacks such as complexity and a user interface that is hard to understand.

Riskly, our proposed system has a fresh UI and is easier to access and understand.

# Requirement Analysis

## Gathering requirements

**Must have requirements:**

* Employee details (name, team, project/assignment)
* Threshold value
* Calendar With Filters (Highlight All Risks, Highlight Today’s Risks, scroll To Today, Add Comments To High Risks)
* Legend.
* Data Entry.
* Risk Chart.
* Resource Wise Chart (Count Of Team Members In Each Team)

**Should have requirements:**

* Colors For Each Member Of A Team

**Nice to have requirements:**

* Public holidays with location
* Categorizing leaves - sick leaves, vacation leaves
* Multiple roles

## 

## High Level Plan

Dates mentioned in green show completed tasks, whereas red dates show expected deadlines.

|  |  |
| --- | --- |
| Week 1 | Requirement Gathering  Deciding Project Scope  Assumptions Made In The Plan : Data is pre-populated via SQL injections.  Design Database  Deliverables : ERD, Sequence Diagram  GUI Design  Design Completed |
| Week 2 | Develop System Modules  Integrate System Modules |
| Week 3 | Perform Unit Testing |
| Week 4 | Perform System Testing  Document Issues Found  Correct Issues Found  Deployment |

Table : High Level Plan

# Use Cases

Use case scenarios describe each process by mentioning actors outside the system as well as specific interactions that they have with it. Each Use Case below defines process goal (desired end result), actor (anyone who interacts with the system), pre-condition (the step previous to the current process), success scenario (what defines whether a process is successful or not), post-condition (what is the status of the system once the current process is completed).

|  |  |
| --- | --- |
| ID | 1 |
| Name | Dashboard |
| Goal | To provide an overview of Employee’s leaves |
| Actor | Employee |
| Pre-condition | User is logged in, and is on the Dashboard section of the web application. |
| Success scenario | 1. Users get an overview of all employees. 2. Users can sort employee lists alphabetically. 3. Users can change calendar view from weekly to monthly. 4. Users can come back to “Today” just with one button click. 5. Users can add new leaves using the floating button. |
| Post-condition | Users can see complete information about leaves. |

Table : Use Cases

|  |  |
| --- | --- |
| ID | 2 |
| Name | Resources |
| Goal | To provide a team-wise view of employees. |
| Actor | Employee |
| Pre-condition | User is in the Resources section of the web application. |
| Success scenario | Users can sort employee lists alphabetically. |
| Post-condition | Users can team-wise employee lists. |

|  |  |
| --- | --- |
| ID | 3 |
| Name | Risk Chart |
| Goal | To provide an overview of team-wise risks. |
| Actor | Employee |
| Pre-condition | User is on the Risk Chart section of the web application. |
| Success scenario | 1. Users can see team-wise risks, depending on team thresholds. 2. Users can understand risk level by reading the legend at the top of the web page. 3. Users can hover over a cell to view more details.. |
| Post-condition | Users can see complete information about leaves. |

# System Analysis & Design

This document provides all the information regarding the design architecture & the components within the Leave Tracker application. The application includes the following components - the classes and the attributes used.

## Database Design

Entity- Relationship Diagram

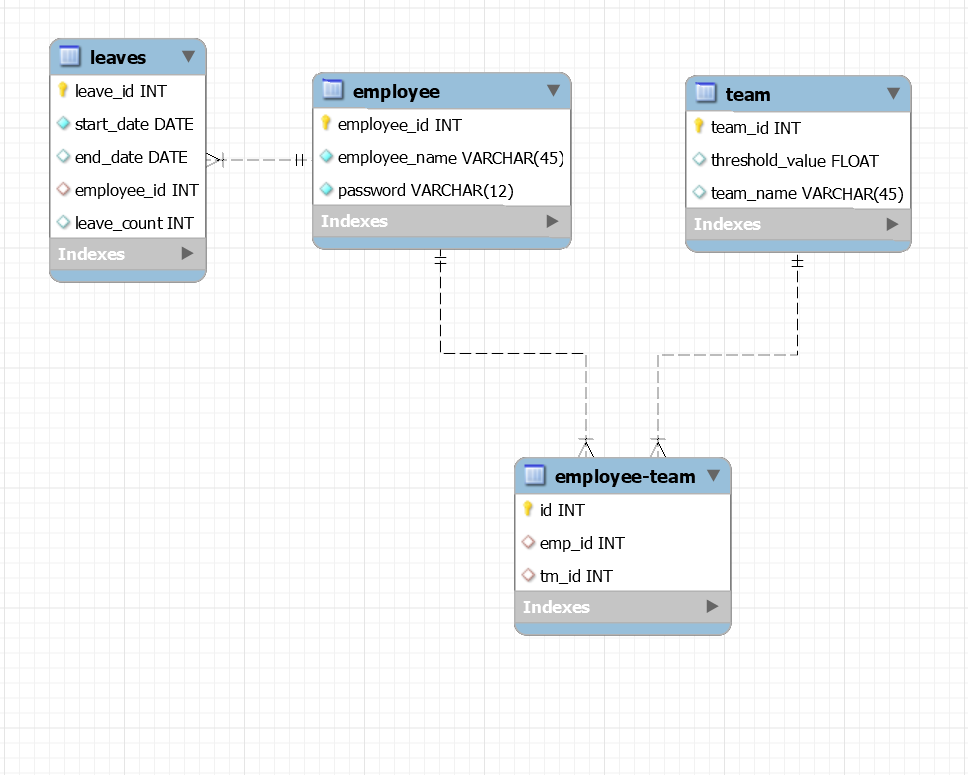


Figure : Entity Relationship Diagram

* This diagram shows us the relationships between entities in our database.
* One employee can apply for multiple leaves, hence it is a one-many relationship.
* One team has many employees and one employee can be assigned to multiple teams. This makes it a many-many relationship. Hence, there exists a separate table to map the relationship between these two entities.

Entity Description

EMPLOYEE

This database table stores employee details, such as name and password. Each row is identifiable via the employee\_id, which is used as primary key and is unique for each employee.

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Constraints |
| employee\_id | INT | PRIMARY KEY |
| employee\_name | VARCHAR(45) | NOT NULL |
| password | VARCHAR(12) | NOT NULL |

Table : Entity Description

TEAM

This table stores the team details, containing the team id, name of the team and its threshold value. Team id is used to uniquely identify each team.

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Constraints |
| team\_id | INT | PRIMARY KEY |
| team\_name | VARCHAR(45) |  |
| threshold\_value | FLOAT | NOT NULL, 0<threshold\_value<=1 |

LEAVE

This database table contains leave information. Each row has an auto generated leave id used as primary key, starting date, end date and number of days of the leave(leave\_count) along with employee id as foreign key which helps us to identify the employee who has applied for this leave.

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Constraints |
| leave\_id | INT | PRIMARY KEY |
| employee\_id | INT | FOREIGN KEY  EMPLOYEE(employee\_id) |
| start\_date | DATE |  |
| end\_date | DATE |  |
| leave\_count | INT | Default value 1 |

EMPLOYEE-TEAM

This table is used to link the employees with their teams. As it is a many to many relationship, both employee\_id and team\_id are foreign keys fetched from table employee and team respectively.

|  |  |  |
| --- | --- | --- |
| Attribute | Data Type | Constraints |
| id | INT | PRIMARY KEY |
| e\_id | INT | FOREIGN KEY EMPLOYEE(employee\_id) |
| t\_id | INT | FOREIGN KEY TEAM(team\_id) |

## 

## Component Diagram

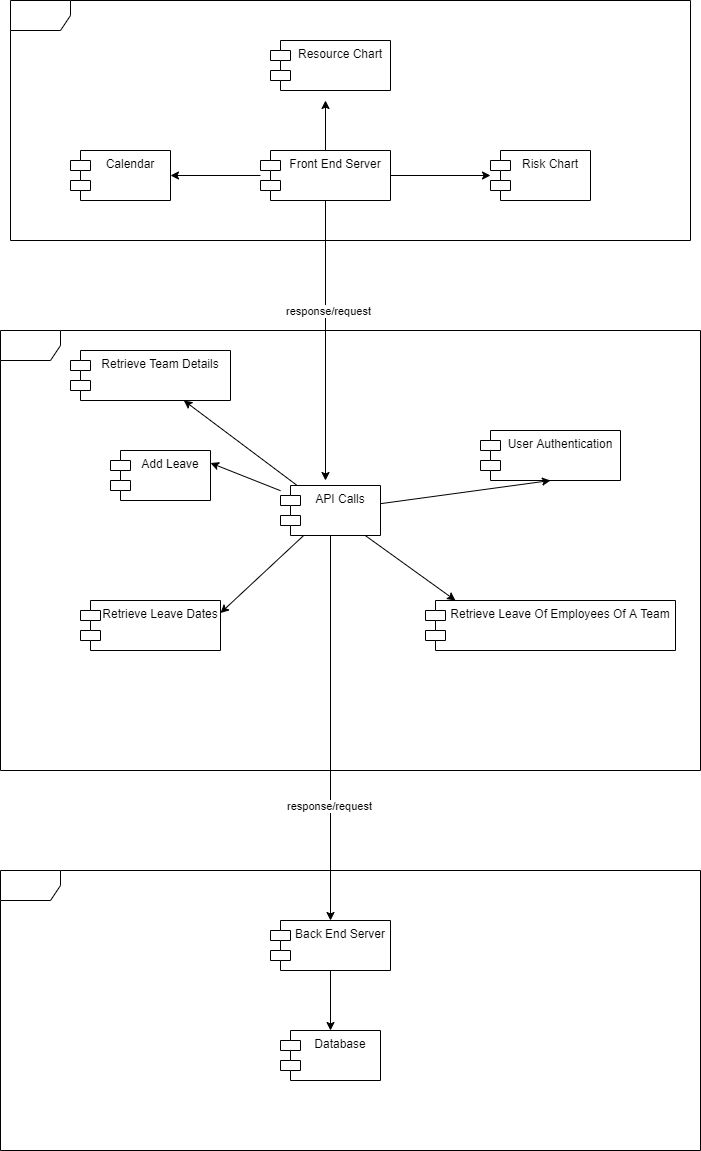


Figure : Component Diagram

## 

## Technology Stack

## 

* Front-end server - ReactJS
* API Calls - ExpressJS & NodeJS
* Back-end server & Database - NodeJS & MySQL

## Libraries used

## 

|  |  |  |
| --- | --- | --- |
| Server Side Libraries | | |
| Library Name | Version | Description |
| chai | 4.3.1 | Writing and running test cases for REST APIs. |
| chai-http | 4.3.0 |
| mocha | 8.3.0 |
| cors | 2.8.5 | Enables middleware to use CORS |
| date-fns | 2.17.0 | Provides tools for manipulating JavaScript dates |
| express | 4.17.1 | Middleware to communicate between client & server |

Table : Server-side Libraries

## 

## 

|  |  |  |
| --- | --- | --- |
| Client Side Libraries | | |
| Library Name | Version | Description |
| material-ui/core | 4.11.3 | Designing UI |
| bootstrap | 4.6.0 |
| react-bootstrap | 1.5.0 |
| react-icons | 4.2.0 |
| create-react-class | 15.7.0 | Creating React components |
| react-scripts | 4.0.2 |
| date-fns | 2.17.0 | Provides tools for manipulating JavaScript dates |
| focus-trap-react | 8.4.2 | To retain focus on Modal & its contents |
| jquery | 3.5.1 | Handling events |
| react-moment | 1.1.1 | Date Pickers inside the Modal |
| react-datepicker | 8.0.7 |
| react-router-dom | 5.2.0 | Routing between pages in the web application |
| react-js-popup | 2.0.4 | Create React popup |
| react | 17.0.1 | Create interactive UIs |

Table : Client-side Libraries

## 

## Sequence Diagram

The following diagram is a visual representation of how processes happen, and what actions trigger them.

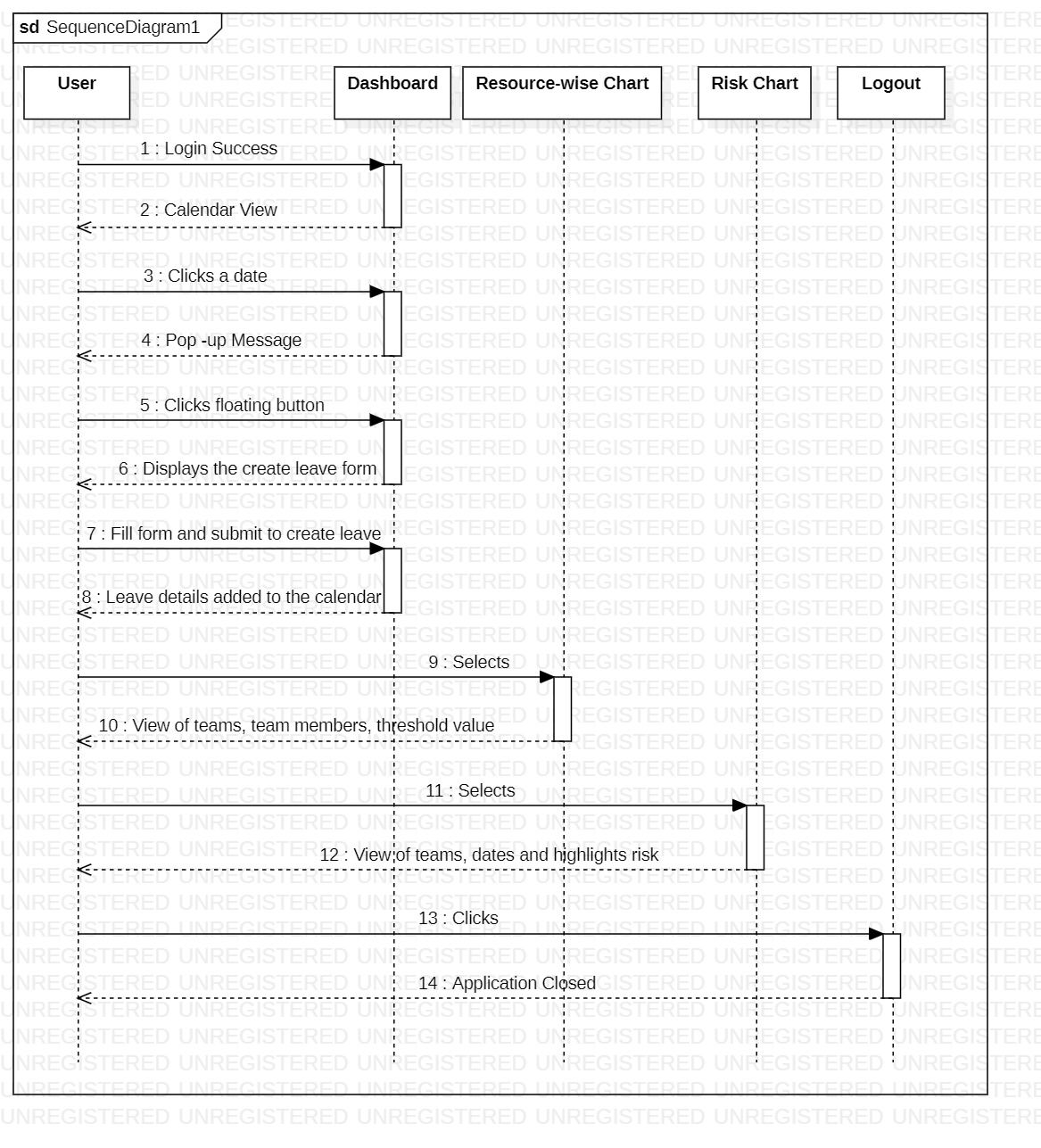


Figure : Sequence Diagram

## Screen Design

The diagrams below are wireframes of the proposed screens in Riskly. They define what elements should be placed in each screen, and what actions can be performed on these elements.

Login

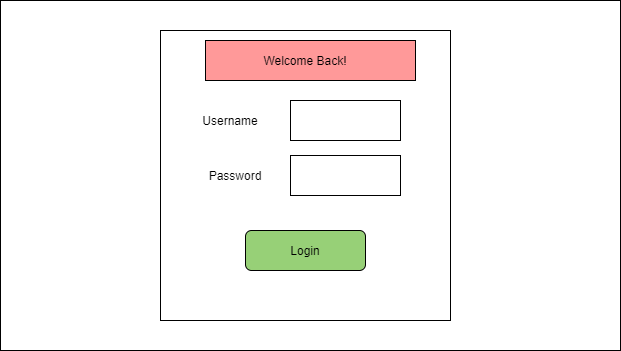
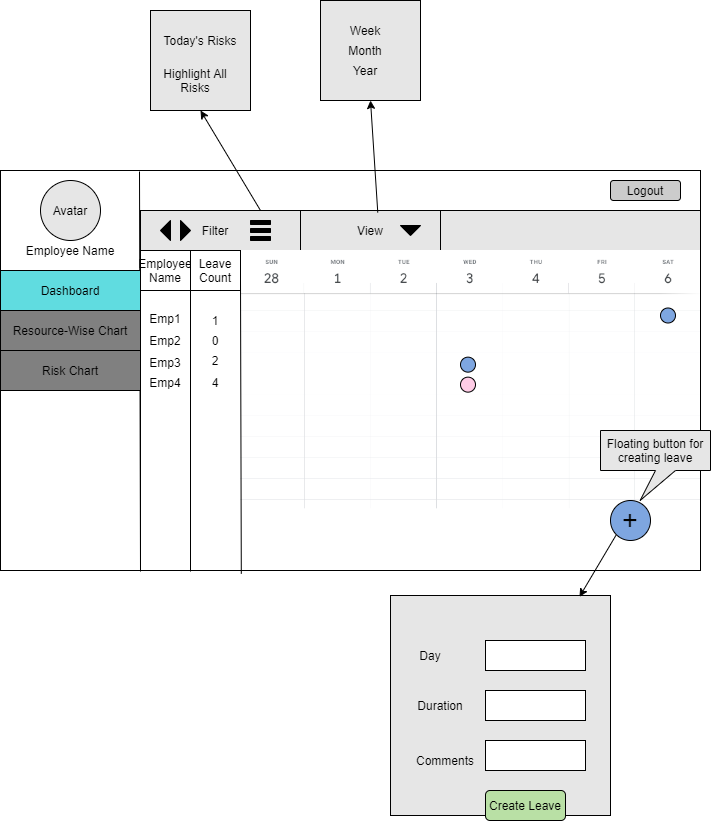


Figure : Wireframes

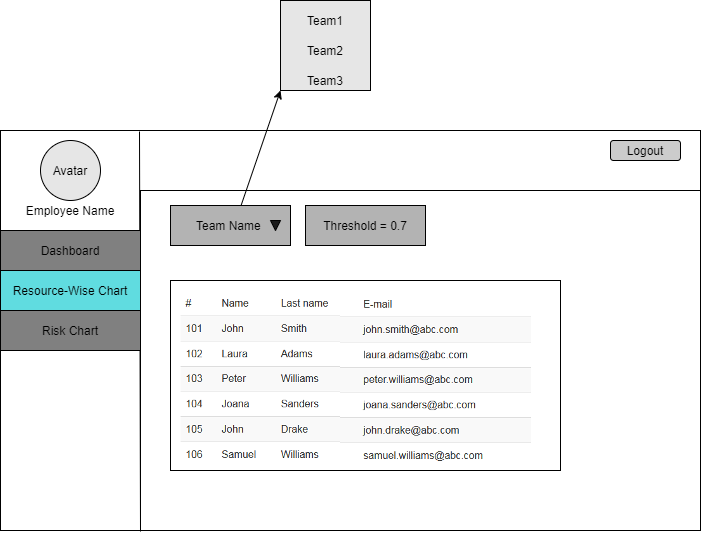
* Login page where employee logs in with pre-assigned ID and password.

Dashboard



* This is the first page that the user lands on after successful login.
* It has a calendar view of the employees on leave, where rows are employee names with respective leave count, and columns are days of the week (by default ).
* Highlighted cells denote the respective employees on leave.
* The toolbar contains 3 options -
* Arrow buttons to navigate back and forth in the calendar.
* Option to filter calendar based on risks.
* Option to toggle calendar view between week, month and year.
* On clicking the floating button, a form pops up where the user can add a leave.
* The user can navigate to other screens using the navbar.
* The user can logout from any screen.

Resource Wise Chart



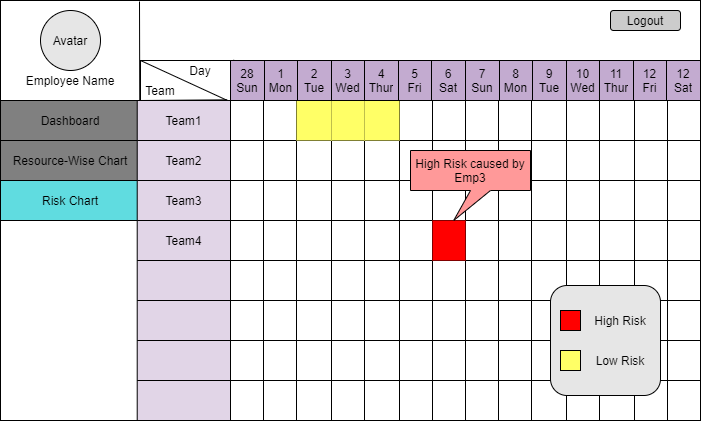
* This is a view of team wise employee data with their respective threshold value.

Risk Chart

* Contains a calendar-like view where rows are team names and columns are days.
* Highlighted cells can be of 2 types -

- Yellow cells indicate that team member(s) are on leave but the threshold value is not crossed.

- Red cells indicate that team member(s) are on leave and the threshold value is crossed.



# Implementation

The screenshots below are actual screens in the web applications. They have been designed by referring to the wireframes in the above section.

### Login Page

User logs in with pre-assigned ID and Password. Without authorization, User cannot progress further.

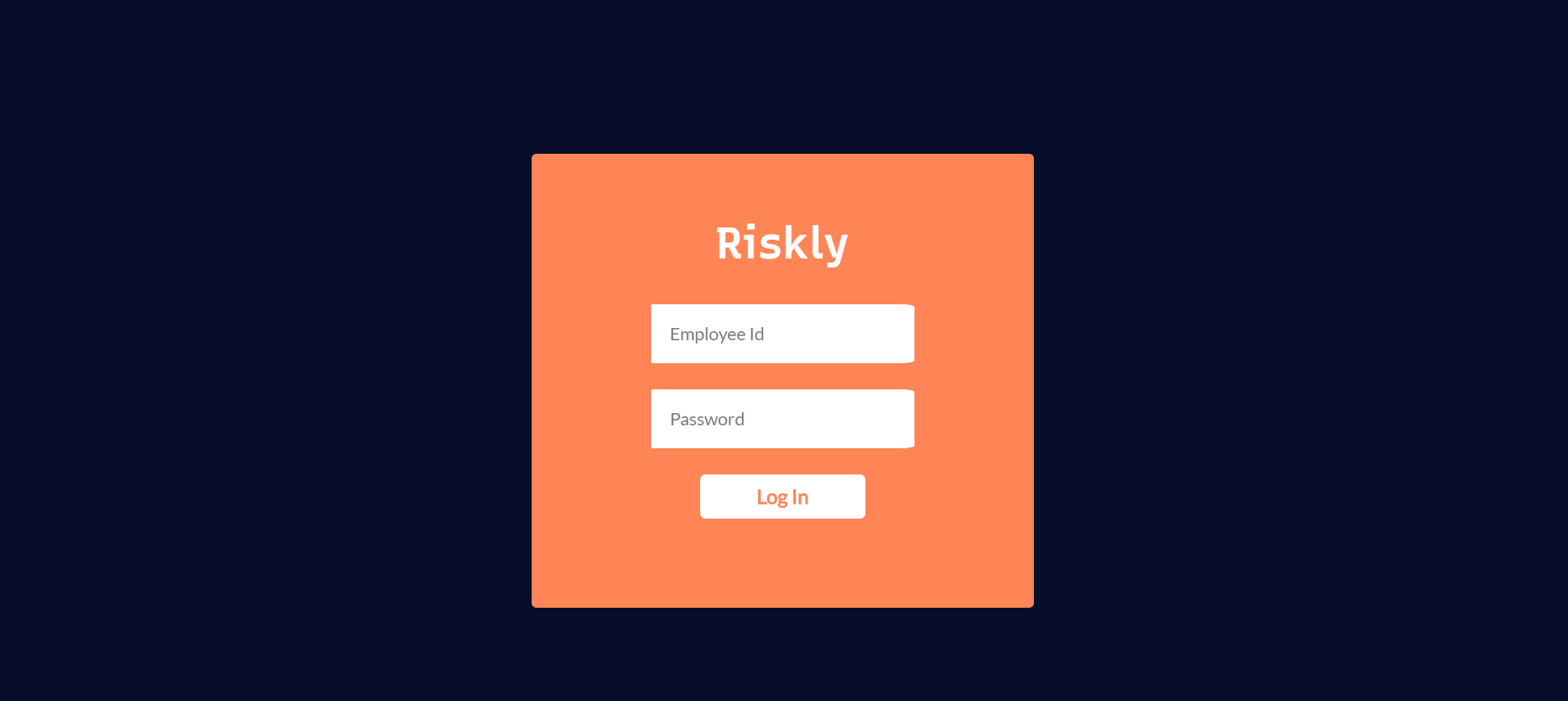
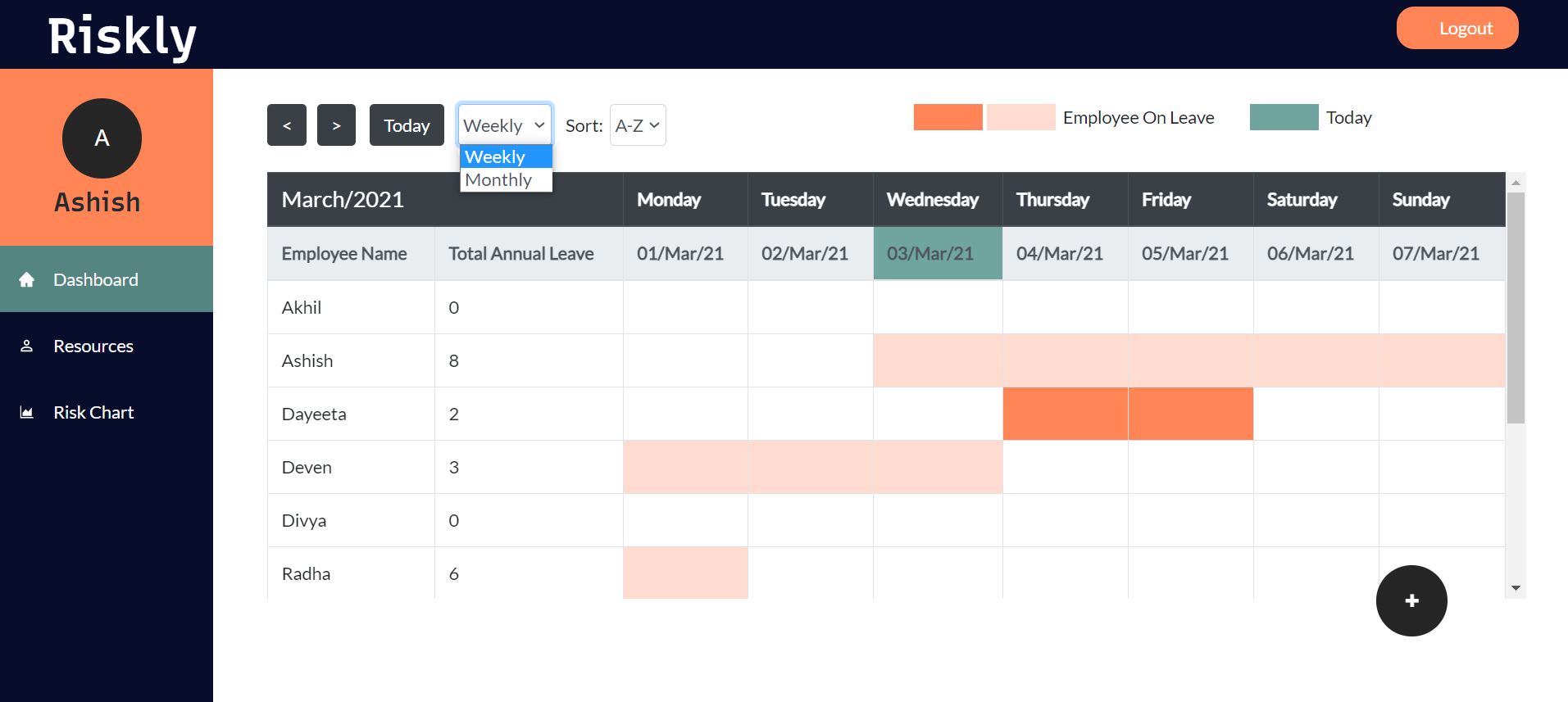


Figure : Screenshots

### Dashboard

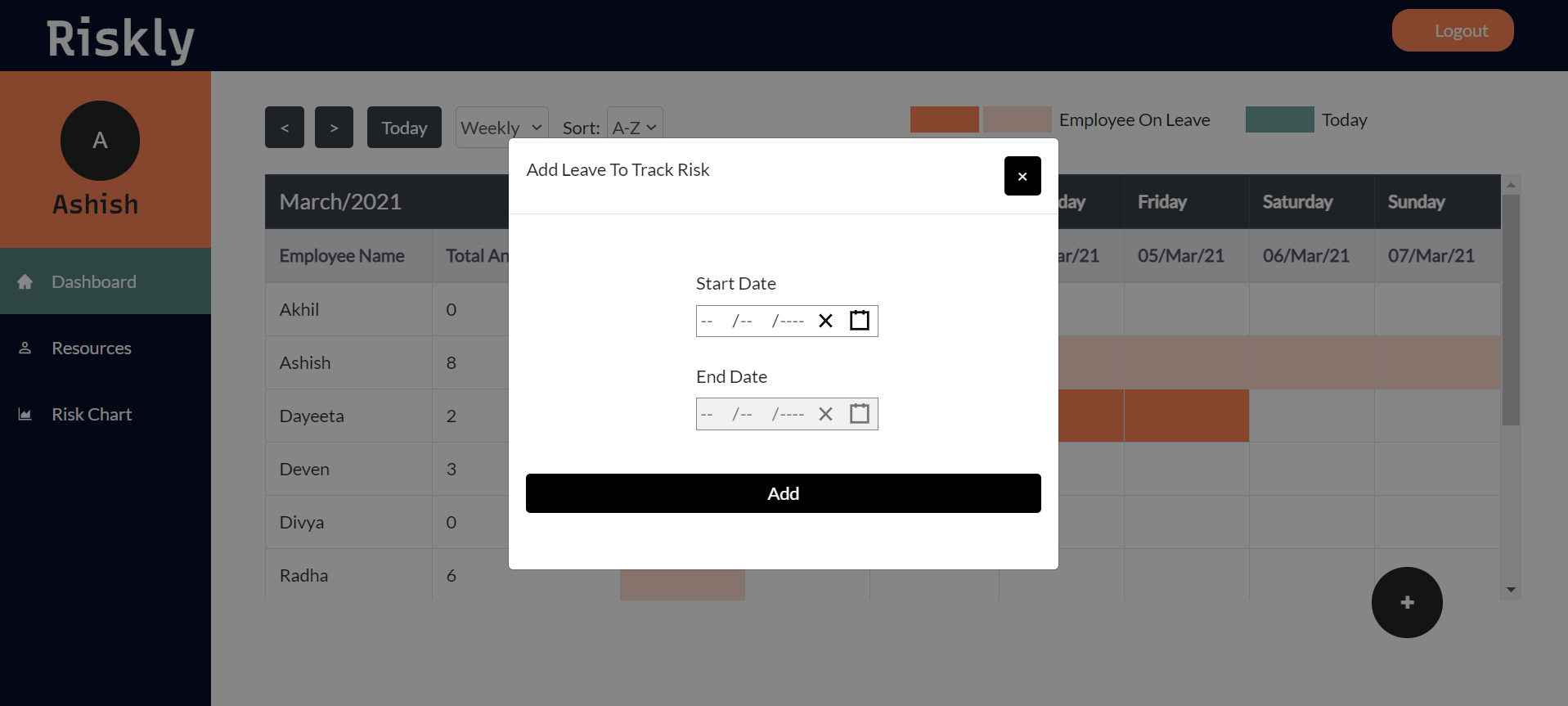
Contains - Employee names and their annual leave count

* Icons to go to previous as well as next week/month
* Icon highlighting today
* Filter calendar for weekly and monthly view
* Drop-down to sort employee names

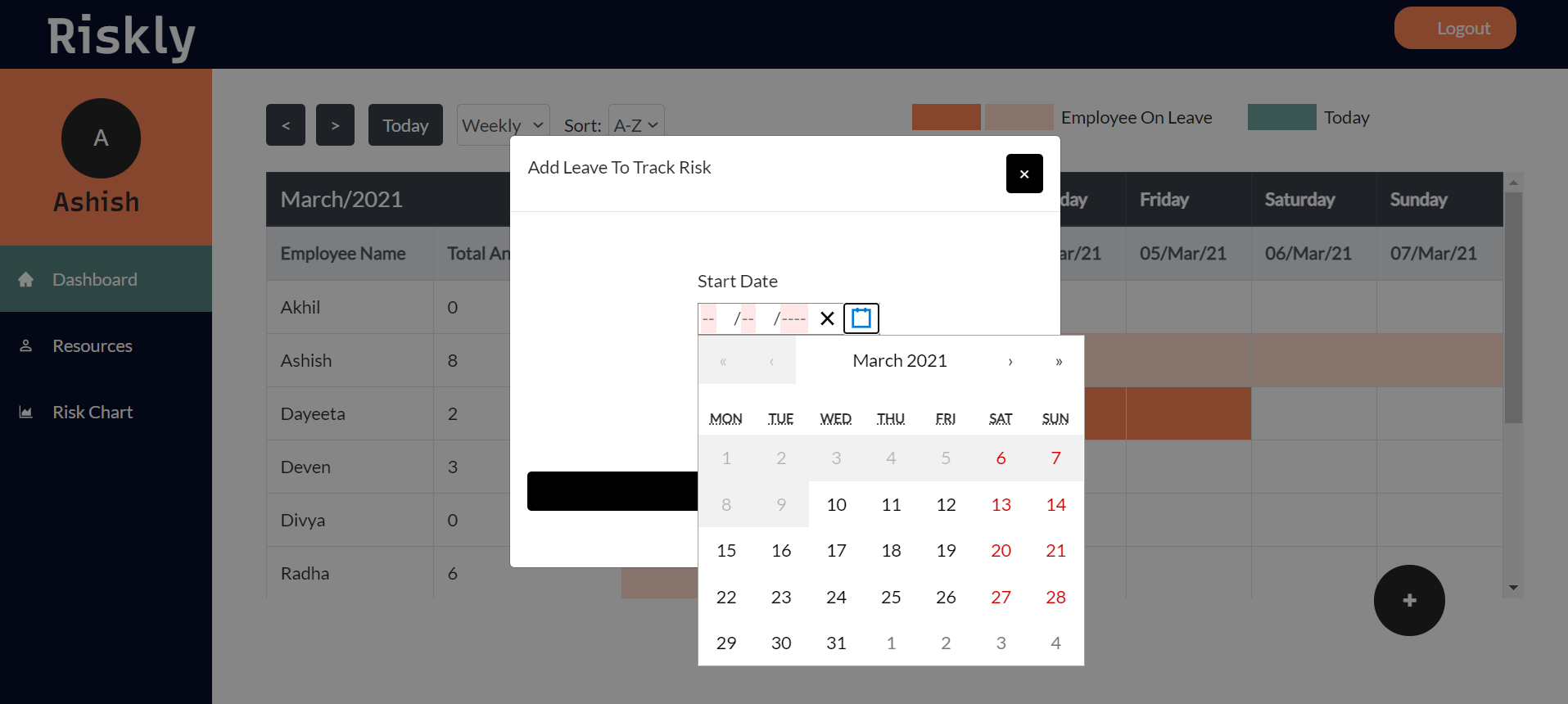


### Add Leave modal

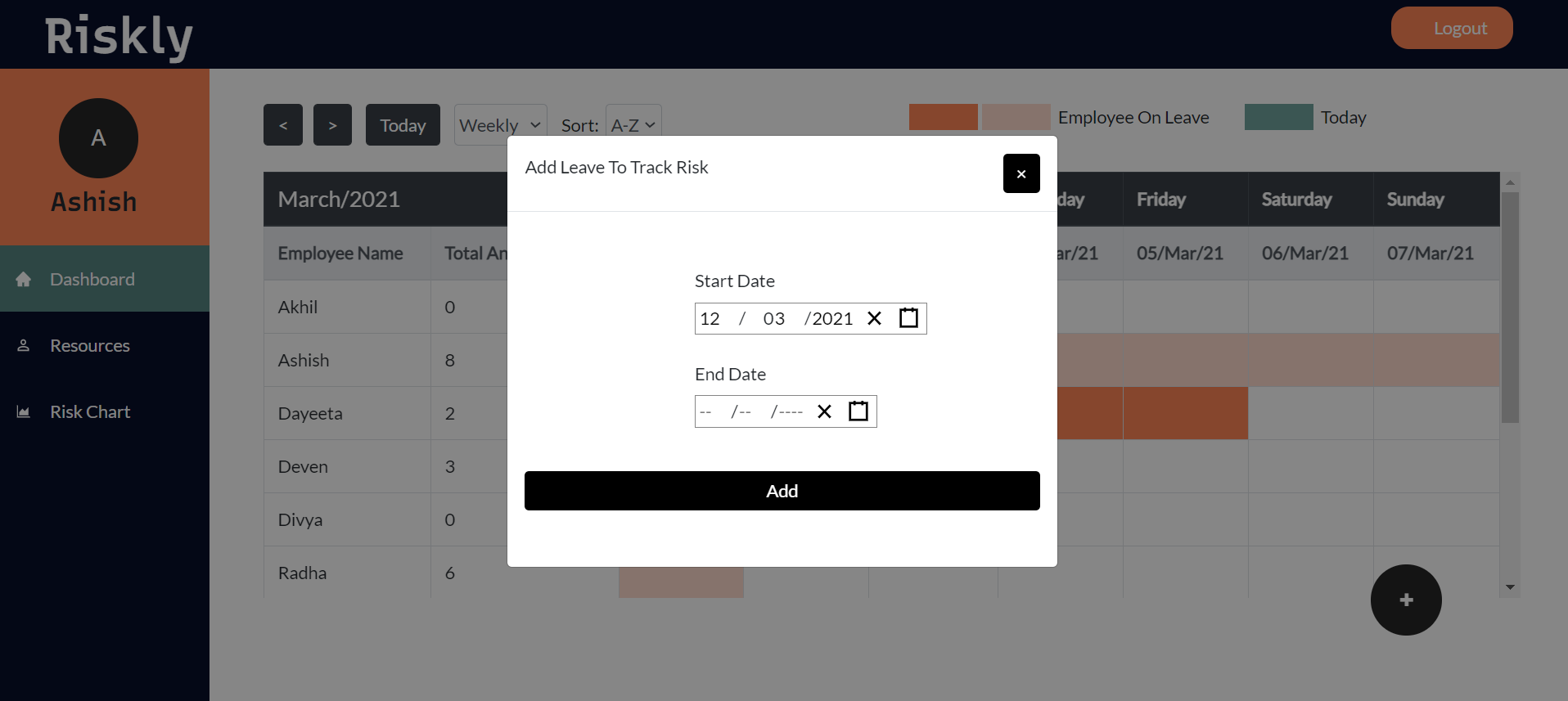
On clicking the floating button, a pop-up form shows up. Here, users can add new leave(s).



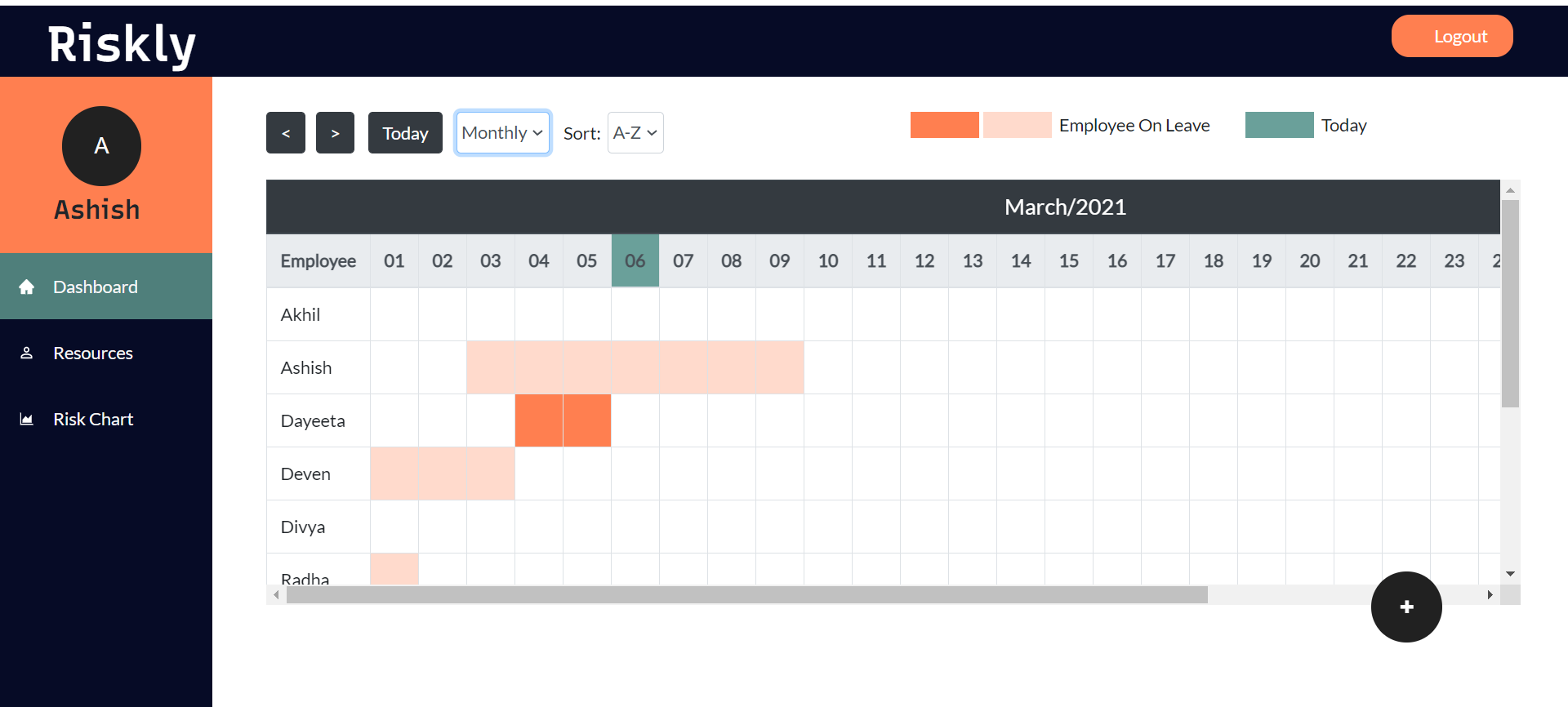
Users can select Start Date for his/her leave.



Once Users have selected Start Date, End Date field is enabled for selection.

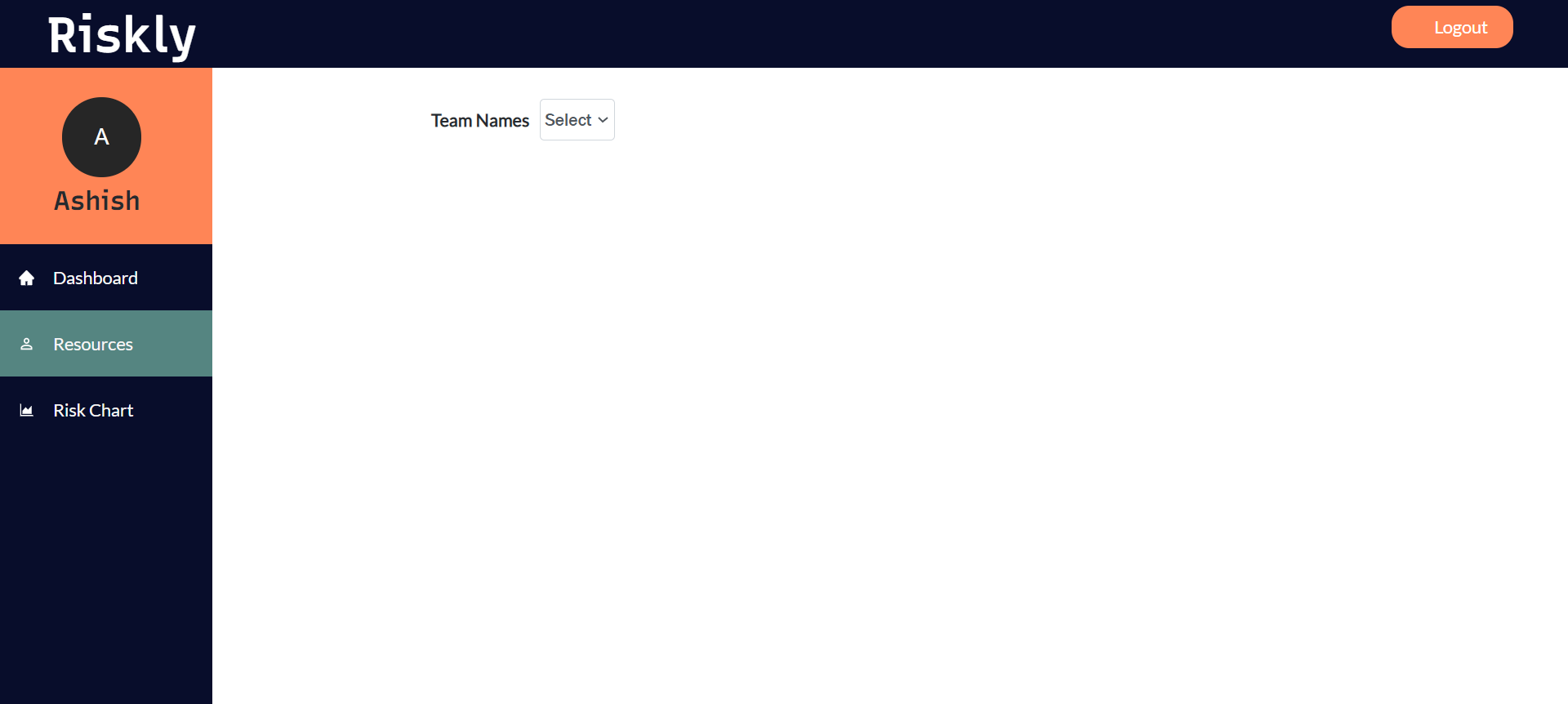


Users can also select a monthly view for Dashboard.

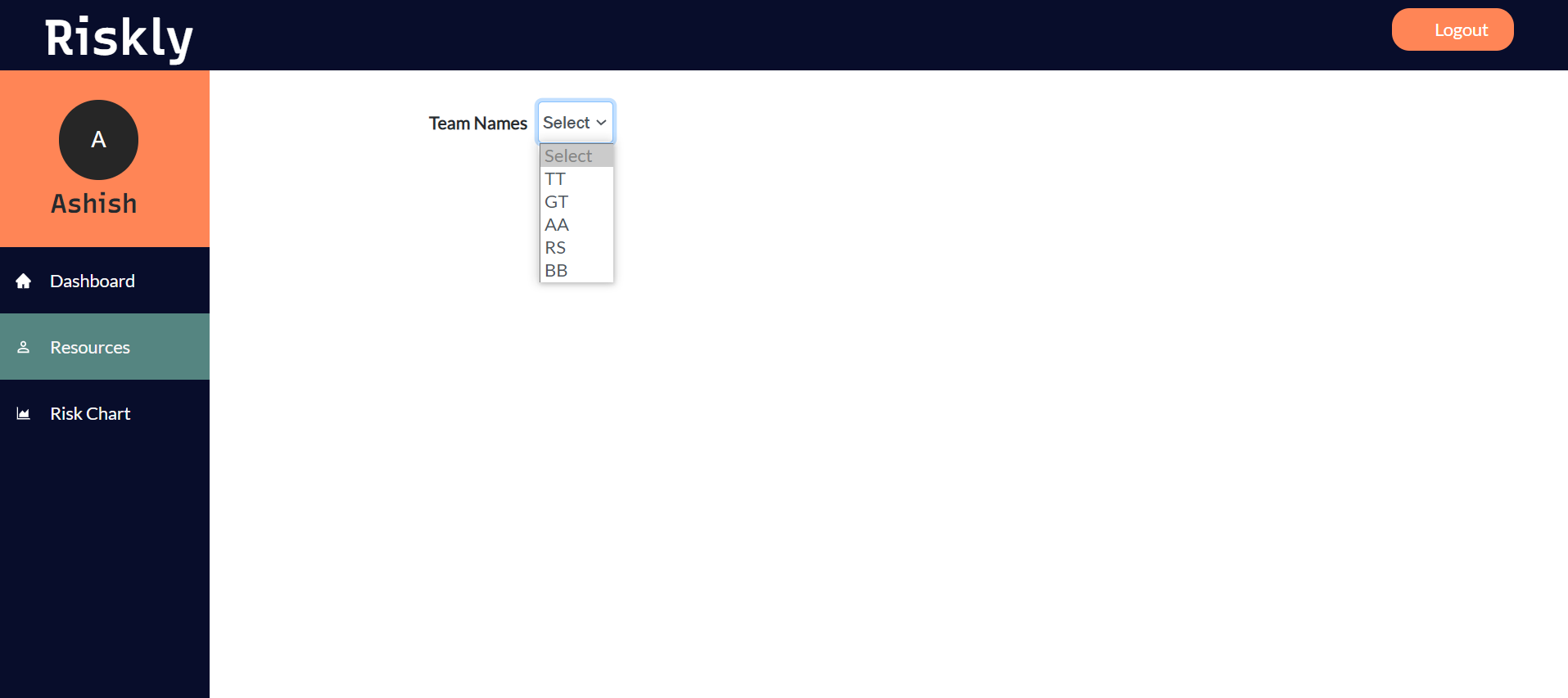


### Resources

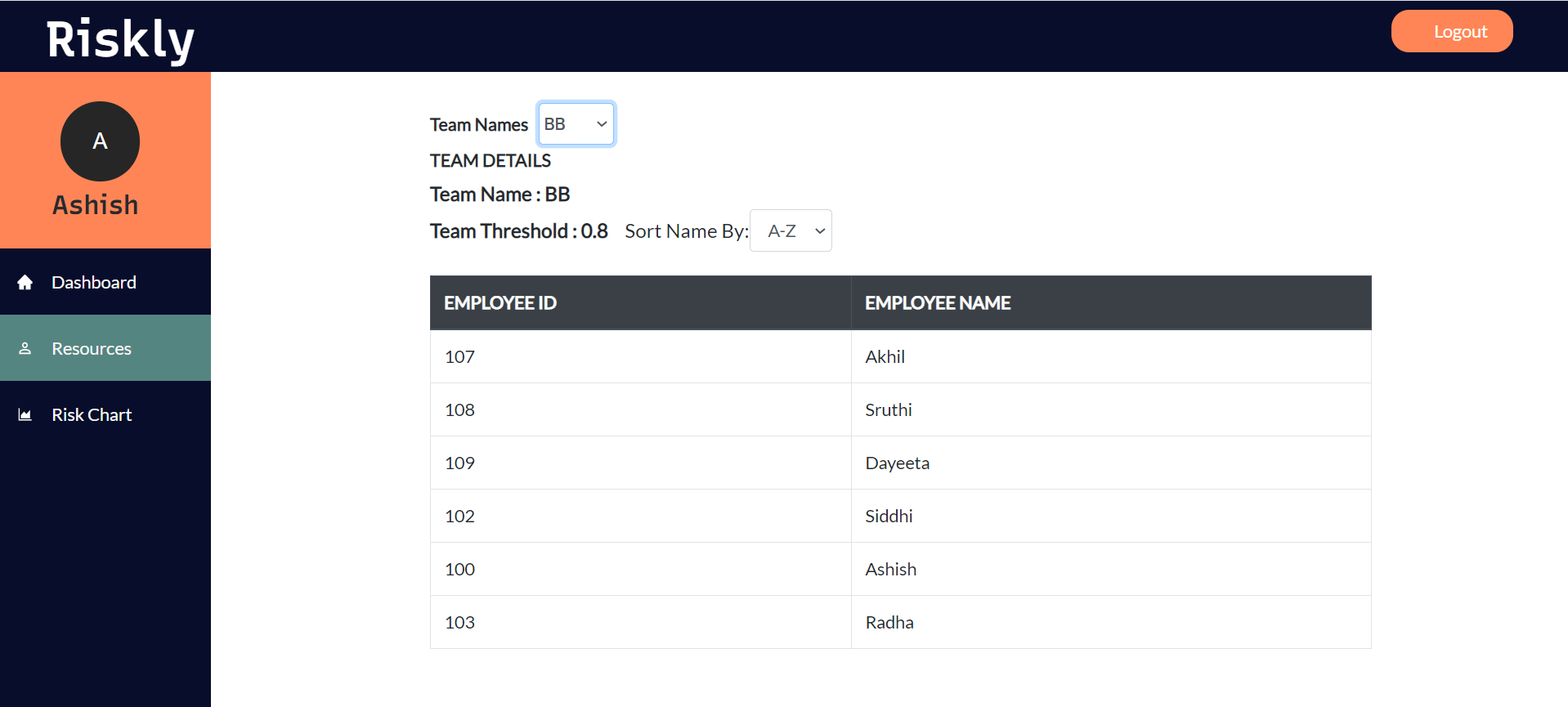
Users can select team names from a drop down list.



Drop-down list contains names of all Teams.



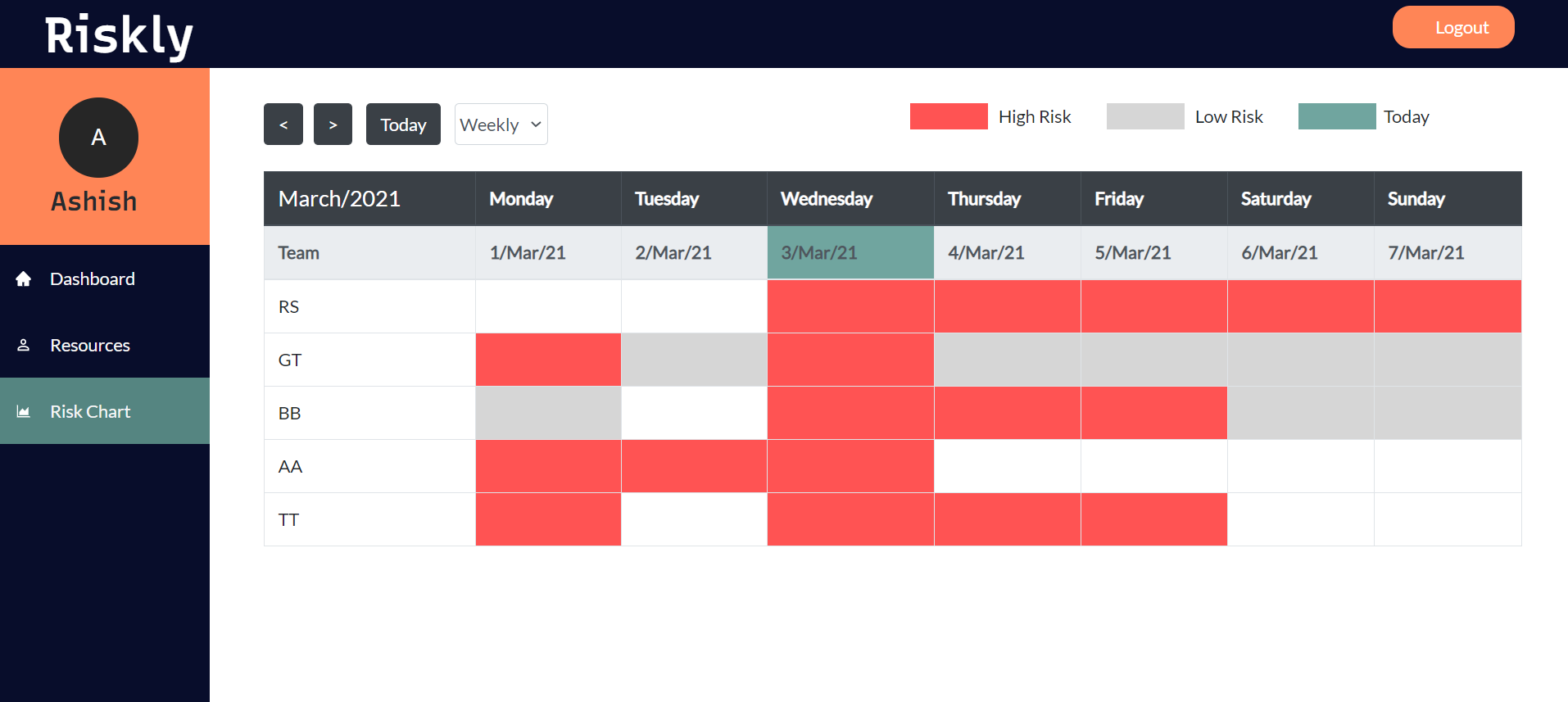
On selecting the team name, the system displays a list of employees in the team. Users can also sort this list alphabetically or in reverse order.



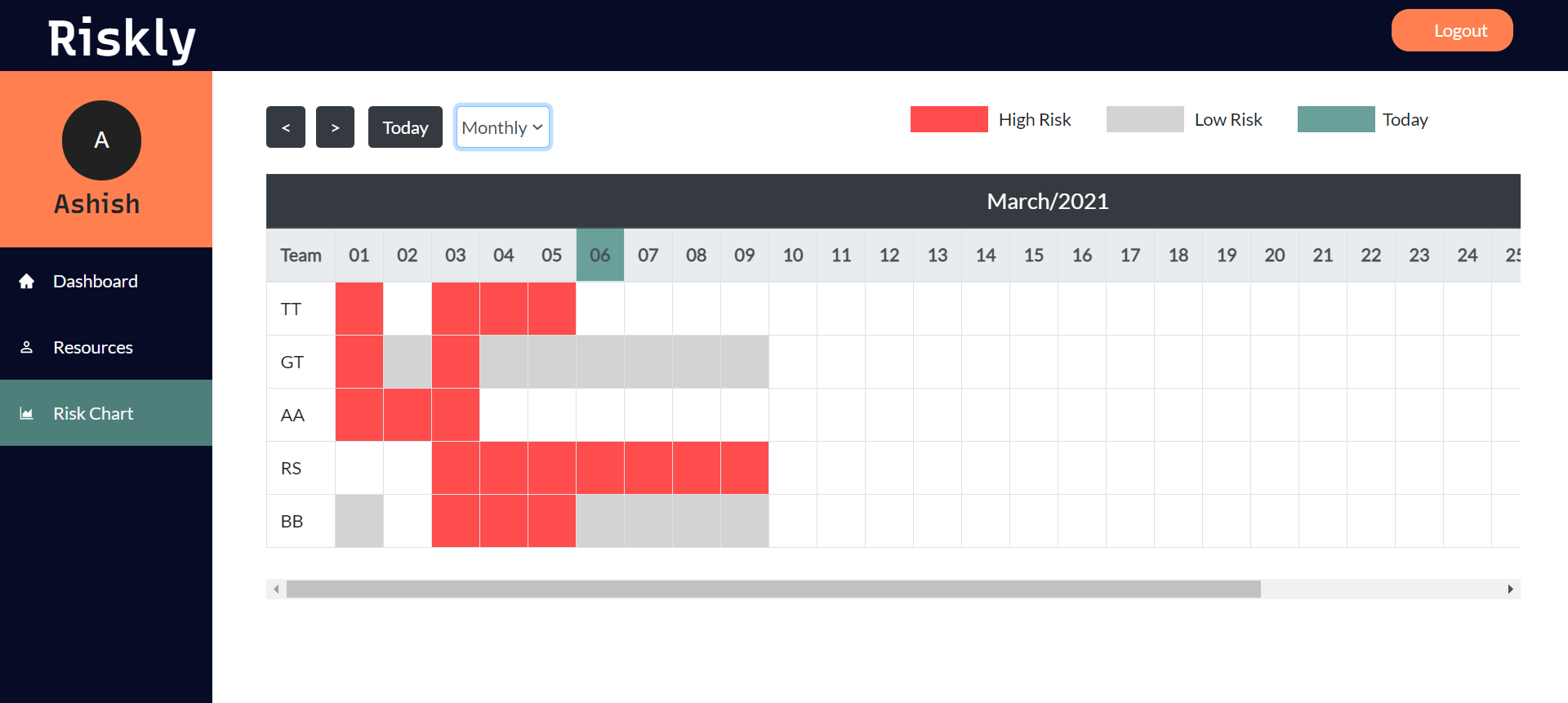
### Risk Chart

Contains -

1. Team names
2. Highlights the risks occurring due to leave of a team member
3. On hovering over the risks of a particular date gives details of the team’s actual threshold as well as current threshold with the team members on leave on that day



Monthly view for Risk Chart.



# Testing

This report gives a summary of the various tests performed and the final test results. It can be used by the clients to evaluate the quality of our application and assess how well the testing is performed.

## Front End Application

### Test Case - UX01

Test case description: Provide a Login page for all employees.

Pre-condition: All employees are allocated username and password.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step No. | Step Description | Test Data | Expected Result | Actual Result |
| 1 | The first page rendered when the application starts is the login page, requiring two inputs which are Employee ID and Password.  User clicks login.  Employee ID and password of the logged in employee is stored in local storage. | 1.Employee ID/Password Not Entered.  2. Incorrect Employee ID  3. Incorrect Password  4. Valid Credentials | System must alert the user about these missing fields.  System must alert the user about the incorrect employee-ID.  System must alert the user about the incorrect password.  Users must be redirected to the dashboard. | System gives an alert stating, "Please fill out this field".  System gives an alert stating, "No such user, check your id or contact Admin".  System gives an alert stating, "Incorrect password, try again".  Redirects the user to the dashboard. |

Table : Test Case - Front-end

### Test Case - UX02

Test case description: Provide a calendar view of employee-wise leaves.

Precondition: User is on the Dashboard section of the web application.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step No. | Step Description | Test Data | Expected Result | Actual Result |
| 1 | User clicks on ‘<’ button on the dashboard.  Prev() displays the previous week/month according to the filter selected. | Current week/month and filter selected. | System displays  previous week/month and filter selected. | System displays  previous week/month. |
| 2 | User clicks on ‘<’ button on the dashboard.  Prev() displays the previous week/month according to the filter selected. | Current week/month and filter selected. | System displays  previous week/month and filter selected. | System displays  previous week/month. |
| 3 | User clicks on the ‘<’ button on the dashboard.  Prev() displays the previous week/month according to the filter selected. | Current week/month and filter selected. | System displays  previous week/month and filter selected. | System displays  previous week/month. |
| 4 | User toggles calendar view filter.  handleChange() sets a state variable according to the filter selected. | Filter value selected. | System switches from weekly view to monthly, and vice versa. | System switches from weekly view to monthly, and vice versa. |
| 5 | User changes Sort filter.  sortEmp() sorts the list of employee objects, according to the filter selected. | Filter value selected and list of employees. | Employee list is sorted in ascending or descending order. | Employee list is sorted in ascending or descending order. |
| 6 | User clicks on ‘Logout’ on the header.  Local Storage is cleared. | Object with current logged in user’s details. | User is logged out of the system. | User is logged out of the system. |

### Test Case - UX03

Test case description: Provide a team-wise view of employees.

Precondition: User is in the Resources section of the web application.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step No. | Step Description | Test Data | Expected Result | Actual Result |
| 1 | User clicks on drop-down list for Team Names.  handleChange() retrieves list of employees in the selected team. | Selected team name. | System displays a list of employees in the selected team.  List is sorted in ascending/ descending order. | System displays a list of employees in the selected team.  List is sorted in ascending/ descending order. |
| 2 | User changes ‘Sort Name By’ filter.  handleSort() sorts employee list. | List of employees in selected team, and filter chosen. | System displays a list of employees in the chosen team, in ascending or descending order, depending on the chosen filter. | System displays a list of employees in the chosen team, in ascending or descending order, depending on the chosen filter. |
| 3 | User clicks on ‘Logout’ on the header.  Local Storage is cleared. | Object with current logged in user’s details. | User is logged out of the system. | User is logged out of the system. |

### Test Case - UX04

Test case description: Provide a team-wise calendar view of risk levels.

Precondition: User is on the Risk Chart section of the web application.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step No. | Step Description | Test Data | Expected Result | Actual Result |
| 1 | User clicks on the ‘<’ button.  Prev() changes current displayed week to previous week. | Current week/month and filter selected. | System displays  previous week/month and filter selected. | System displays  previous week/month. |
| 2 | User clicks on the ‘<’ button.  Prev() changes current displayed month to next month. | Current week/month and filter selected. | System displays  previous week/month and filter selected. | System displays  previous week/month. |
| 3 | User clicks on the ‘<’ button.  Prev() displays the previous week/month according to the filter selected. | Current week/month and filter selected. | System displays  previous week/month and filter selected. | System displays  previous week/month. |
| 4 | User toggles calendar view filter.  handleChange() sets state variables according to the filter selected. | Filter value selected. | System switches from weekly view to monthly, and vice versa. | System switches from weekly view to monthly, and vice versa. |
| 5 | User changes Sort filter.  sortEmp() sorts the list of employee objects. | Filter value selected and list of employees. | Employee list is sorted in ascending or descending order. | Employee list is sorted in ascending or descending order. |
| 6 | User hovers over the calendar cell.  getToolTipList() returns required data. | Day from selected cell (column) and team name (row). | System displays tooltip with employee on leave along with respective team name and thresholds. | System displays tooltip with employee on leave along with respective team name and calculated vs. expected thresholds. |
| 7 | User clicks on ‘Logout’ on the header.  Local Storage is cleared. | Object with current logged in user’s details. | User is logged out of the system. | User is logged out of the system. |

### Test case - UX05

Test case description: Redirect to appropriate page when user enters invalid address.

Precondition: -

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step No. | Step Description | Test Data | Expected Result | Actual Result |
| 1 | User tries to access a valid address without logging in. |  | User is redirected to the Login page. | User is redirected to the Login page. |
| 2 | User enters invalid address in the URL.  Router redirects user to ‘/404’ path. |  | User is redirected to the Error Page. | User is redirected to the Error Page. |

## Back-End Application

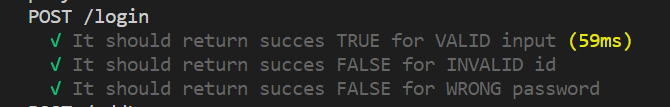
### Test Case - BackEnd01

Test case description: Get employee's login details (employee ID, password).

Precondition: Server is running.

|  |  |
| --- | --- |
| Step Description | User provides the employee ID and password and clicks login. |
| Test Data | Employee ID: 100  Password: 123 |
| URL Under Test | http://localhost:3000/login |
| Expected Result | The system allows the user to successfully login. |
| Actual Result | The system verifies the entered credentials, if valid allows the user to access the dashboard. |

Table : Test Case - Backend

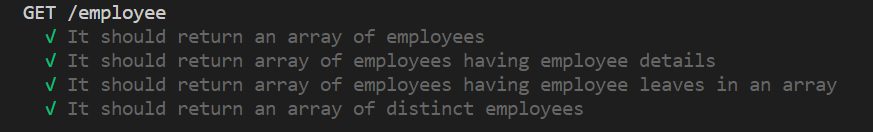


### Test Case - BackEnd02

Test case description: Get employee name and leave count for all employees.

Precondition: Server is running

|  |  |
| --- | --- |
| Step Description | User is at the dashboard/clicks dashboard. |
| Test Data | none |
| URL Under Test | http://localhost:3000/employee |
| Expected Result | The system displays a calendar view of all employees with their leave count. |
| Actual Result | The system displays all employees with their annual count of leaves taken and the range of leave dates highlighted. |

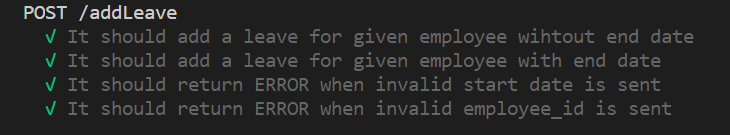


### Test Case - BackEnd03

Test case description: Add leave for the employee logged in.

Precondition: Server is running

|  |  |
| --- | --- |
| Step Description | User provides the start date (mandatory) and end date to add leave. |
| Test Data | Start Date : 8/03/2021  End Date : 10/03/2021 |
| URL Under Test | http://localhost:3000/addLeave |
| Expected Result | The system displays these days as leaves in the calendar. |
| Actual Result | The system checks whether the end date is null or not, if null the leave count is assigned 1 and if not null the leave count is calculated and added as leave in the calendar for the logged in employee. |

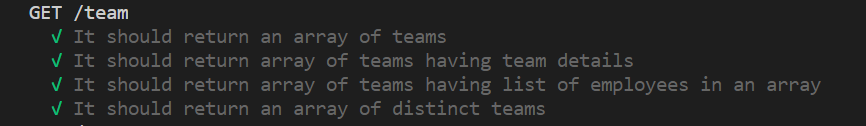


### Test Case - BackEnd04

Test case description: Get team-wise details.

Precondition: Server is running

|  |  |
| --- | --- |
| Step Description | User clicks resources and selects the required team from the drop down list of which details are to be fetched. |
| Test Data | TT-Technical |
| URL Under Test | http://localhost:3000/team |
| Expected Result | The system displays team-wise details. |
| Actual Result | The system displays a summary of the required team with details such as team name, team threshold and employees assigned to that team. |

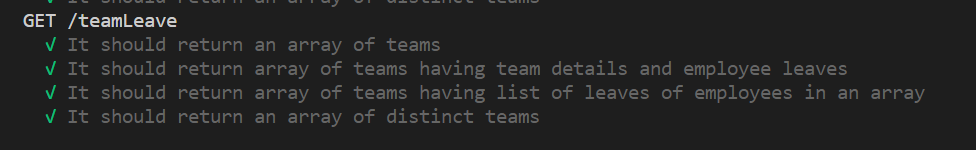


### Test Case - BackEnd05

Test case description: Get details of the risk condition occurring in each team.

Precondition: Server is running

|  |  |
| --- | --- |
| Step Description | User clicks risk-chart. |
| Test Data | none |
| URL Under Test | http://localhost:3000/teamLeave |
| Expected Result | The system displays team wise details of employees on leave and highlights the risk conditions. |
| Actual Result | The system displays all teams and highlights the risk condition with details of team threshold, current threshold and employees on leave on a particular day as a calendar view |



# Limitations

* System does not manage dynamic data. All data is pre-populated.
* System does not consider public holidays as default leaves.
* System does not have a separate role for higher authority like manager etc.

# Future Enhancements

* Consider different locations and their public holidays.
* Separate role for higher authority, and one for Admin.
* Categorize leaves.

# Conclusion

Our system, Riskly, provides a more attractive UI which is easier to understand. It is easy to navigate, which makes it user friendly.

Previous system was static, whereas Riskly uses a database. This makes it easier to perform data operations and is dynamic in nature.

The calendar view in Riskly gives information at a glance. It shows simplified views of important data such as employee leaves, leave count, teams, employees assigned to a team and teams-at-risk. Teams at risk are highlighted with bright colors to make it noticeable.

The above reasons contribute to making it easier for the user to analyze risks caused by his/her leave.

Check out our Git repository:

[divya-kulkarni/Leave-Tracker: Mini Project 01 - at BluePineapple (github.com)](https://github.com/divya-kulkarni/Leave-Tracker)

References

* [wojtekmaj/react-date-picker: A date picker for your React app. (github.com)](https://github.com/wojtekmaj/react-date-picker)
* [Express 4.x - API Reference (expressjs.com)](http://expressjs.com/en/4x/api.html)
* <https://reactjs.org/docs/>
* [date-fns - modern JavaScript date utility library (date-fns.org)](https://date-fns.org/docs/Getting-Started)
* [Usage - Material-UI (material-ui.com)](https://material-ui.com/getting-started/usage/)
* [React-Bootstrap · React-Bootstrap Documentation (react-bootstrap.github.io)](https://react-bootstrap.github.io/components/alerts)
* [mysqljs/mysql: A pure node.js JavaScript Client implementing the MySQL protocol. (github.com)](https://github.com/mysqljs/mysql)
* [Introduction - Chai (chaijs.com)](https://www.chaijs.com/api/)
* [React Router: Declarative Routing for React.js](https://reactrouter.com/web/guides/quick-start)