

**Headcount**

**Allocation**

**Project**

Application Requirements Document

Nofar Cohen Sedek

Gili Veltz

Noa Malul

Hadas Printz

Version: 1.0

Table of Contents

Introduction .…………………………………………………………………………………. 3

* Vision …………………………………………………………………………….……. 3
* Stakeholders ………………………………………………………………….…….. 4
* Software context………………………………………………………………..….. 5

User Scenarios …………………………………………………………………………...... 8

* User Profile …………………………………………………………………………… 8
* Manager Use cases ……………………………………………………………... 12
* Employee use cases ……………………………………………………….……. 18
* Big Manager use cases …………………………………………………………. 21

Functional Requirements ……………………………………………………………… 22

* Emploee and project management ………………………………………… 22
* Issue Tracking ……………………………………………………………………… 22
* Gap Analysis and Resolution …………………………………………………. 23
* Interactive Scheduling ………………………………………………………….. 23
* Reporting and Visualization……………………………………………………. 23
* User Access and Security………………………………………………………. 23
* Notifications and Alerts…………………………………………………………. 24
* Recommendation Feedback………………………………………………….. 24

Non-Functional Requirements ………………………………………………………. 25

* Performance……………………………………………………………………….. 25
* User Interface (UI/UX) ………………………………………………………….. 25
* Data Security ………………………………………………………………………. 25
* Accessibility ……………………………………………………………………….. 25
* Scalability ………………………………………………………………………….. 26
* Availability ………………………………………………………………………….. 26

Risk Assessment …………………………………..…………………………………….. 27

* Technical risks ……………………………………………………………………. 27
* Operational risks …………………………………………………………………. 27
* Project risks ………………………………………………………………………… 27
* Legal and Compliance Risks …………………………………………………. 27

Proof of Concept (PoC) Plan ………………………………………………………….. 29

Introduction

Vision**:**  
Headcount allocation is an intuitive and advanced system for managing teams and projects. Its purpose is to provide team managers with intelligent tools for effective human resource and task management. The system enables rapid problem identification, management of complex scenarios, and tailored solutions to streamline daily operations and enhance collaboration between managers and employees.

The Problem We Aim to Solve:  
In many teams, project management requires precise alignment between project needs and employee characteristics. Each project is unique in its nature, technical, and operational requirements, necessitating the assignment of the right people to the right tasks.

## Examples of Common Challenges:

* Unique Requirement Matching:  
  Projects involving international teams require employees available during flexible hours, such as night shifts, and fluent in high-level English. Similarly, projects utilizing specific technologies like Python demand employees with expertise in these areas.
* Handling Sudden Staff Shortages:  
  Unexpected employee absences due to events like military service, maternity leave, or extended vacations pose significant challenges to teams. Managers often lack optimal solutions to fill the vacancy, potentially affecting project progress and quality.

## Our System's Solution:

* The system analyzes project and employee characteristics to suggest intelligent matches. For instance, in the event of an absence, it identifies the best candidate for substitution based on criteria such as availability, professional knowledge, and language proficiency, prioritizing criteria according to project needs.
* The system assists team managers in optimally assigning employees while considering the unique requirements of each project and team dynamics.
* It enables managers to visualize team structures and project options for each assignment and alerts them to potential issues in other teams caused by the changes.
* This approach ensures swift and accurate responses to unforeseen problems, enhances operational efficiency, and improves team productivity while saving managers valuable time.

# Stakeholders:

## Team Managers

* Role:  
   Enter employee and project data, manage task and project progress, and assign employees according to their abilities.
* Needs from the System:
  + Clear visualization of team and project status in real-time.
  + Quick identification of operational issues, such as resource shortages and employee availability changes.
  + Efficient problem-solving through tailored recommendations.
  + Automated suggestions for employee assignments.

## Team Members

* Role:   
  Access proposed solutions for team issues and input personal preferences (e.g., language, working hours, task type, and upcoming leave).
* Needs from the System:
  + Personal status display within projects.
  + Updated information about team changes and updates.
  + Inputting personal constraints.

## Organization Leadership

* Role:  
   Oversee team and project operations to ensure alignment with organizational goals.
* Needs from the System:
  + Access to strategic data such as resource utilization, timeline adherence, and performance metrics.
  + Long-term trend analysis based on system data.
  + Improved organizational efficiency and risk reduction.

## Development Teams (Internal or External)

* Role:  
  Develop, maintain, and improve the system per user and stakeholder requirements.
* Needs from the System:
  + Clear specification of requirements, processes, and issues for continuous improvement.
  + Ongoing feedback from team managers and employees for optimal adjustments.

# Software Context:

System Purpose:  
The system aims to provide a centralized platform for team and resource management, enabling real-time issue identification and resolution, resource optimization, and tailored solutions for evolving needs.

## Key Functionality:

1. Team and Project Management:
   * Input employee lists, including personal characteristics (working hours, preferred language, programming language proficiency, availability).
   * Track project statuses, including performance metrics and resource utilization rates.
   * Display an overall status view: active employees, open projects, and current resource utilization.
2. Issue Management:
   * Open "tickets" in the system to report problems such as employee absences (leave, military service, etc.).
   * A smart mechanism presenting options to resolve the issue (e.g., available employees matching required traits).
   * Select solutions and display team status after implementing the solution.
3. Resource Planning and Management:
   * Assign employees to tasks based on defined criteria (language, programming skills, working hours).
   * Display forecasts and implications for each decision (e.g., how assignments affect other teams and projects).
4. Advanced Interactive User Interface (UI):
   * A dashboard displaying real-time data visually.
   * Filtering and customizing information to meet user needs.
   * User-friendly experience for both managers and employees.

## Integrations and Technological Infrastructure:

* **Infrastructure:** The system relies on modern, cloud-based infrastructure to ensure high availability and data security.
* **Data Security:** Compliance with advanced security standards to ensure data privacy.

## Potential Users:

* Organizations with dynamic teams requiring real-time adjustments.
* Technology companies, professional services, and projects with fluctuating work cycles.

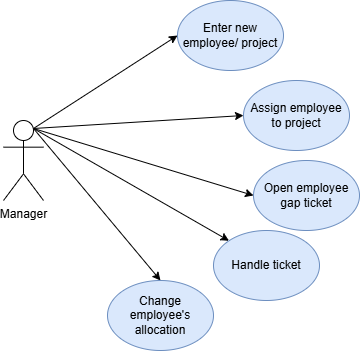
User scenarios

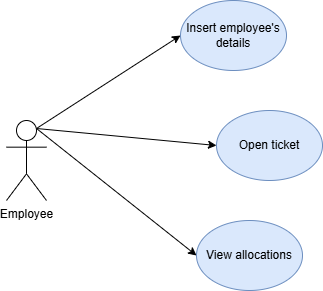
# User profile:

The system users are project managers / human resource managers. Technological skills: good knowledge of project management tools. Works in a mid-to-large company with multiple on-going projects and employees (even projects and employees around the world).

In some of those companies' employees work on more than one project simultaneously.

Some users will have good computer skills, but some users will be far from the computer world, so the system must be very user friendly.





תמונה שמכילה קו, תרשים, עיגול, עיצוב

התיאור נוצר באופן אוטומטי

# Manager Use - Cases

## Enter a new employee/project:

### Description:

The manager enters a new employee/ project to the system- with all its attributes.

### Actor:

* The manager.

### Pre- conditions:

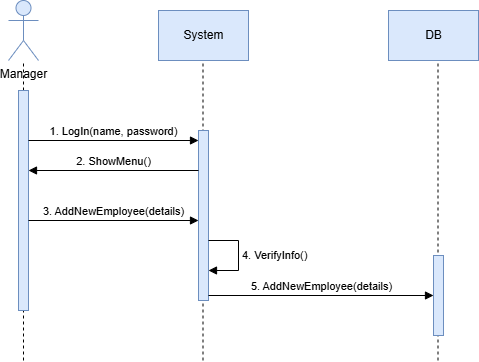
* The manager has access to the system, and is logged in.
* The employee/ project is not in the system.
* All the attributes are correct and logical (hours, age…)

### Post- conditions:

* The new employee/ project is in the system.
* The manager can see the new employee/ project and assign it.

### Main scenario:

1. The manager logs in to the system.
2. The manager selects the option of adding new employee/ project.
3. The manager enters all the requested attributes.
4. The system checks for correctness of the fields.
5. The system saves the new employee/ project and displays it to the manager.

****

## Assign employee to project:

### Description:

The manager assigns employee to existing project.

### Actor:

* The manager.

### Pre- conditions:

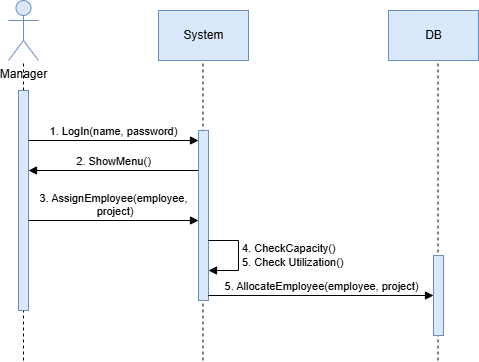
* The manager has access to the system, and is logged in.
* The employee and project are in the system.
* The employee's utilization is suitable for the project's requirements.

### Post- conditions:

* The employee is assign to the project
* The employee's utilization updates and the project's requirements also.

### Main scenario:

1. The manager logs in to the system.
2. The manager selects the option of assigning employee to a project.
3. The manager enters the relevant employee and project.
4. The system checks for employee's utilization and project's capacity and approve the assignment.
5. The system saves the new assignment and displays it to the manager.

****

## Open employee gap "ticket"

### Description:

The manager open a ticket of employee's absence.

### Actor:

* The manager.

### Pre- conditions:

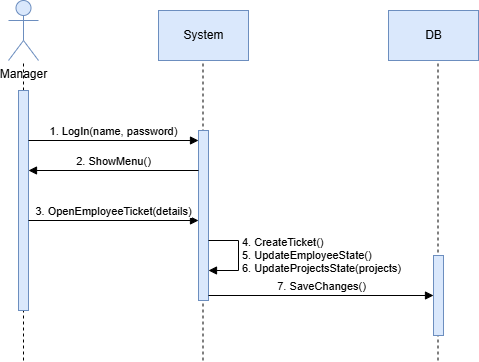
* The manager has access to the system, and is logged in.
* The employee is in the system.

### Post- conditions:

* The ticket is in the system and visible to the manager.
* The gaps in the projects where the employee is assigned are visible to the manager in the gaps page.

### Main scenario:

1. The manager logs in to the system.
2. The manager selects the option of opening new employee's absence ticket.
3. The manager enters the relevant employee, dates and reason of leaving.
4. The system creates gaps tickets for each project the employee is assign to, with details of the gaps.

****

## Handle ticket:

### Description:

The manager closes ticket by changing the assign of the employees between the projects.

### Actor:

* The manager.

### Pre- conditions:

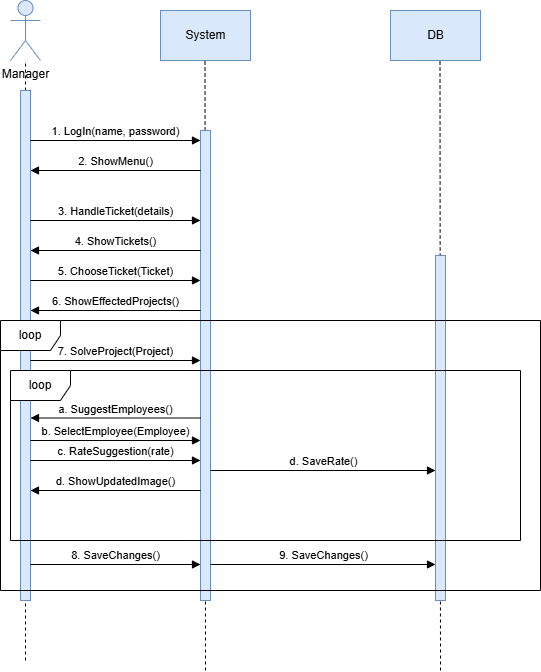
* The manager has access to the system, and is logged in.
* A gap ticket is open and visible to the manager.

### Post- conditions:

* Optional: the ticket is closed.
* Optional: new gap ticket is created.
* Changes of employees assignments are updated in the system and visible to the manager.

### Main scenario:

1. The manager logs in to the system.
2. The manager selects the option of view gaps tickets.
3. The system presents all the gaps tickets.
4. The manager selects option of solving a ticket for a specific ticket.
5. The system shows all the projects effected from the gap according to their priorities.
6. The manager goes into each project and enter the priorities of the project's attributes.
   1. The system suggests employees to fill the gaps based on the priorities of the places with gaps in the project.
   2. The manager chooses the right employee and assigns it to the specific part of the project.
   3. The manager rates the suggestion of the system.
   4. The systemsaves the rate in the DB.
   5. The system updates the change and shows the manager update image of the employee's assignment, projects gaps and utilization.
7. Do a-c until the manager saves the changes.



## Change employee's allocation:

### Description:

The manager changes his employee's allocation between projects manually.

### Actor:

* The manager.

### Pre- conditions:

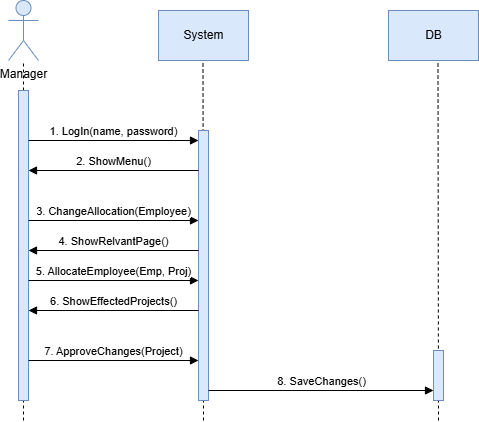
* The manager has access to the system, and is logged in.
* The relevant employee and project are existing in the system and visible to the manager.

### Post- conditions:

* Changes of employee's assignments are updated in the system and visible to the manager.

### Main scenario:

1. The manager logs in to the system.
2. The manager selects the option of changing employee allocation.
3. The manager chooses the relevant employee and allocates it to the new project.
4. The system shows the effect of the actionon the employee and all the projects.
5. The manager approves the changes.
6. The system updates the new allocation.

****

# Employee Use - Cases

## Insert Employee details:

### Description:

The employee enters his details to the system- with all its attributes.

### Actor:

* The employee

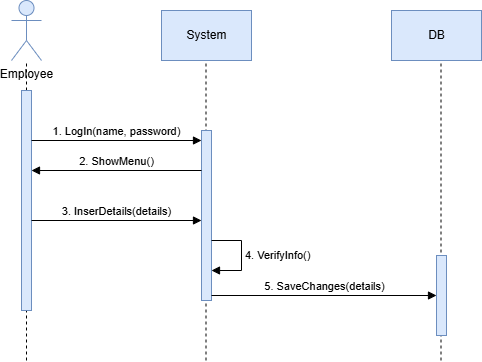
### Pre- conditions:

* The employee has access to the system, and is logged in.
* The employee is not in the system.
* All the attributes are correct and logical (hours, age…)

### Post- conditions:

* The employee's details are in the system.
* The manager can see the new employee/ project and assign it.

### Main scenario:

1. The employee logs in to the system.
2. The employee selects the option of inserting employee's details.
3. The employee enters all the requested attributes.
4. The system checks for correctness of the fields.
5. The system saves the new details and displays them to the manager.

## Open ticket:

### Description:

The employee opens a ticket for his absence.

### Actor:

* The employee.

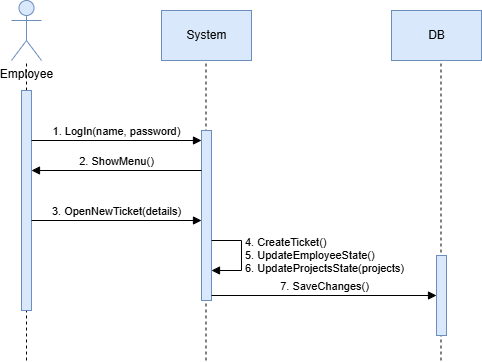
### Pre- conditions:

* The employee has access to the system, and is logged in.
* The employee's details are in the system.

### Post- conditions:

* The ticket is in the system and visible to the manager.
* The gaps in the projects where the employee is assigned are visible to the manager in the gaps page.

### Main scenario:

1. The employee logs in to the system.
2. The employee selects the option of opening a new absence ticket.
3. The employee enters the relevant details such as dates and reason for leaving.
4. The system creates gaps tickets for each project the employee is assigned to, with details of the gaps.

## View employee's allocation to projects:

### Description:

The employee views his allocation to projects- days of week, language, and percentages of work.

### Actor:

* The employee.

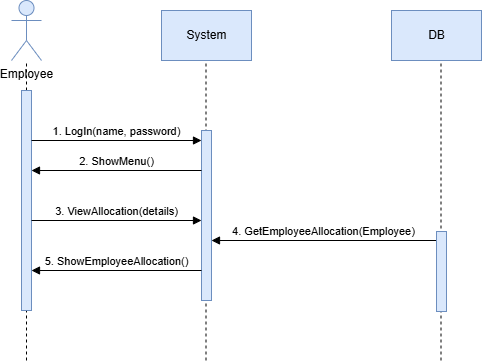
### Pre- conditions:

* The employee has access to the system, and is logged in.
* The employee's details are in the system.

### Post- conditions:

* The employee gets a page with all the projects he is allocated to with all their details.

### Main scenario:

1. The employee logs in to the system.
2. The employee selects the option of view his current allocation status.
3. The system loads a page with all the employee's allocated projects and details.

# Big Management Use - Cases

## View statistics:

### Description:

The manager views details about all the employees, the working hours for each project and more relevant statistics.

### Actor:

* The Big Manager.

### Pre- conditions:

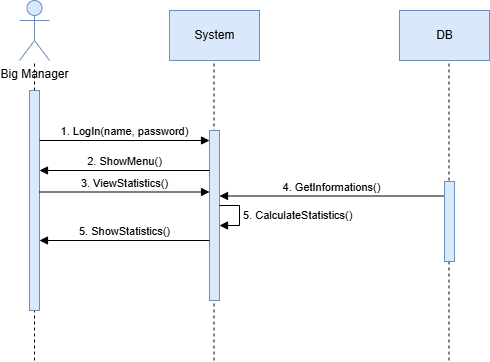
* The Big Manager has access to the system, and is logged in.

### Post- conditions:

* The Big Manager gets a page with all the relevant statistics.

### Main scenario:

1. The Big Manager logs in to the system.
2. The Big Manager selects the option of view statistics.
3. The system loads a page with all the relevant statistics.

****

Functional requirements

## 1. Employee and Project Management

* 1. The system shall allow the team manager to input details for all team members, including:
* Working hours
* Programming languages
* Specific technical skills
  1. The system shall allow the team members to input details of:
* Working hours
* Programming languages
* Upcoming vacations and constraints
  1. The system shall enable the manager to define projects with attributes such as:
* Required hours
* Technology stack
* Assigned team members
  1. The system shall display an updated status board showing:
* Current employee utilization
* Resource allocation summaries

## 2. Issue Tracking

* 1. The system shall include an issue creation feature, enabling managers to log issues like:
* Employee leave (vacation, reserve duty, etc.)
* Project delays
  1. Each issue shall allow input fields for:
* Problem description
* Start and end dates
* Associated employees and projects

## 3. Gap Analysis and Resolution

* 1. The system shall analyze resource gaps caused by logged issues and provide:
* Alternative team members to fill the gap based on predefined attributes (e.g., hours, skills, language compatibility).
* Scenarios showing how changes impact overall project status.
  1. Employees shall have an interface to input their availability and preferences for addressing gaps.

## 4. Interactive Scheduling

* 1. The system shall provide a graphical user interface (GUI) that allows the manager to:
* Adjust employee allocations interactively.
* Simulate project outcomes based on different employee assignments.
  1. The interface shall provide real-time feedback on:
* Updated project timelines
* Employee workload balances

## 5. Reporting and Visualization

* 1. The system shall generate reports and dashboards that include:
* Weekly employee workload summaries
* A visual representation of resource gaps and resolutions applied
  1. Reports shall be exportable in common formats (e.g., PDF, Excel). \*

## 6. User Access and Security

* 1. The system shall provide role-based access control:
     1. Managers shall have full access to all features.
     2. Employees shall have restricted access, limited to:
* Viewing their project assignments and schedules
* Submitting availability and preferences
  1. All data shall be securely stored to protect sensitive employee and project information.

## 7. Notifications and Alerts

* 1. The system shall notify employees and managers via email or in-app notifications about:
* Assigned tasks and changes in schedules
* Pending issues and deadlines
  1. Configurable alerts shall be available for critical updates, such as:
* Resource shortages
* Project delays

## 8. Recommendation Feedback

* 1. After selecting one, The system shall allow the manager to provide ratings for the proposed solutions based on their effectiveness.
  2. Ratings shall be stored in the system to refine future recommendations by prioritizing solutions with higher ratings.

Non-Functional Requirements

## Performance:

* 1. Critical screens, such as team status or reports, must load within 2 seconds under a load of up to 20 concurrent users.
  2. The system must handle up to 50 simultaneous requests without performance degradation
  3. The system must ensure no data loss during an unplanned shutdown and recover to full functionality within 5 minutes.

## User Interface (UI/UX):

* 1. The interface must be responsive and compatible with modern browsers (Chrome, Firefox, Edge).
  2. All key actions must be achievable within a maximum of 5 clicks from the main screen.
  3. Dashboards and reports should feature intuitive graphical elements, including charts, metrics, and easy-to-use filters.

## Data Security:

* 1. User authentication must be implemented using Single Sign-On (SSO) or a secure token mechanism.
  2. Sensitive data, such as employee names and working hours, must be stored in a secure Database.
  3. Access to administrative features (e.g., adding new employees) must be restricted based on user roles and permissions.

## Accessibility:

* 1. Features such as text enlargement and high-contrast mode must be available.

## Maintenance and Upgrades:

* 1. The codebase must be well-documented to allow new developers to onboard quickly.

## Scalability:

* 1. The system must scale linearly and support teams of up to 1,000 employees without performance loss.
  2. Adding new projects, teams, and requirements should not require significant changes to the system architecture.

## Availability:

* 1. The system must operate with 99.9% uptime throughout the year.
  2. Planned maintenance must be communicated to users at least 48 hours in advance.

Risk Assessment

## 1. Technical Risks

* Risk: Performance degradation with increasing number of users.
  + Mitigation: Implement stress testing during development to identify bottlenecks early. Optimize database queries and caching mechanisms.
* Risk: Incompatibility with modern browsers or mobile devices.
  + Mitigation: Conduct cross-browser and device compatibility testing during development.
* Risk: Security vulnerabilities in sensitive data handling (e.g., encryption, authentication).
  + Mitigation: Follow best practices for secure coding, including regular security audits and penetration testing.

## 2. Operational Risks

* Risk: Resistance from users due to unfamiliarity with the system.
  + Mitigation: Provide training materials and user-friendly documentation. Conduct workshops for managers and employees.
* Risk: Errors in data input (e.g., incorrect employee or project details).
  + Mitigation: Implement input validation and error-handling mechanisms in the system.

## 3. Project Risks

* Risk: Delays in meeting deadlines due to scope creep or underestimated complexity.
  + Mitigation: Define clear deliverables for each phase, and use Agile methodology for incremental progress and continuous feedback.
* Risk: Dependency on third-party libraries or tools causing integration issues.
  + Mitigation: Choose reliable, well-documented libraries and maintain backups or alternatives.

## 4. Legal and Compliance Risks

* Risk: Non-compliance with data protection laws if handling employee data.
  + Mitigation: Ensure compliance by consulting legal experts and implementing strict data protection measures.

Proof of Concept (PoC) Plan

# The Alpha version of the project will include:

1. Employee Registration:  
   The system will provide the capability to add a new employee, including essential information such as their skills, available working hours, languages spoken, and other relevant attributes.
2. Project Creation:  
   Users will have the ability to create new projects within the system, defining project details such as name, description, timeline, and associated goals.
3. Role Assignment within Projects:  
   Managers can assign specific roles to a project. Each role will include detailed requirements and skill sets needed for the role, as well as the relative importance or urgency of each skill required for the role.
4. Role Allocation Suggestions:  
   When a role is assigned to a project, the system will offer potential candidates for the role assignment. The manager will be able to review these suggestions and decide whether to accept the suggested allocation or choose an alternative candidate.
5. Employee Input for Constraints and Skills:  
   Employees will have the ability to update and enter their own personal constraints and additional skills into the system, ensuring that the data reflects their evolving availability and capabilities.
6. Dashboard Display for Managers and Team Members:  
   Both managers and team members will have access to a customized dashboard that displays real-time information about the team's current status, ongoing projects, team member involvement in each project, and more. This will allow stakeholders to monitor progress, availability, and team performance at a glance.
7. Data Storage in Database:  
   All user information, project details, and employee-to-project assignments will be securely stored in a centralized database, ensuring that the data is easily retrievable, up-to-date, and consistent across the system.
8. Development Stack:

* Frontend: React, or Vaadin for the user interface.
* Backend: Python (Django/Flask), or Java (Spring Boot) for API and logic.
* Database: SQLServer, PostgreSQL or MongoDB for data storage.