

**Headcount**

**Allocation**

**Project**

Application Requirements Document

Nofar Cohen Sedek

Gili Veltz

Noa Malul

Hadas Printz

Version: 1.4

Table of Contents

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

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

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

* User Profile …………………………………………………………………………… 8
* Manager Use cases ……………………………………………………………... 10
* Employee use cases ……………………………………………………….……. 20
* Big Manager use cases …………………………………………………………. 25

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

* Employee 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
* Scalability ………………………………………………………………………….. 25

Risk Assessment …………………………………..…………………………………….. 26

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.

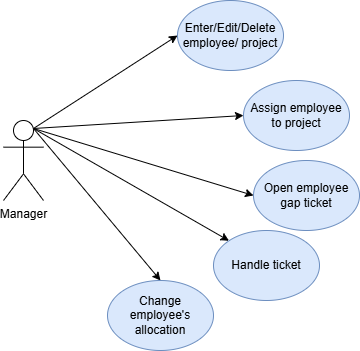
User scenarios

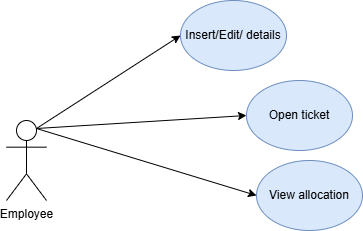
# User profile:

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 טקסט, קו, עיגול, תרשים

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.

**תמונה שמכילה טקסט, צילום מסך, תרשים, קו

התיאור נוצר באופן אוטומטי**

## Edit employee/project:

### Description:

The manager edits the attributes of an existing employee/ project.

### Actor:

* The manager.

### Pre- conditions:

* The manager has access to the system, and is logged in.
* The employee/ project is in the system.
* All the attributes are correct and logical (hours, age…)

### Post- conditions:

* The details are updated.

### Main scenario:

1. The manager logs in to the system.
2. The manager selects the option of editing employee/ project.
3. The manager selects the relevant employee/ project.
4. The manager edits all the relevant attributes.
5. The system checks for correctness of the fields.
6. The system saves change and displays it to the manager.

תמונה שמכילה טקסט, צילום מסך, תרשים, מקביל

התיאור נוצר באופן אוטומטי

## Delete a new employee/project:

### Description:

The manager deletes employee/ project.

### Actor:

* The manager.

### Pre- conditions:

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

### Post- conditions:

* The new employee/ project is deleted.
* New gaps according to the delete are updated and visible to the manager.

### Main scenario:

1. The manager logs in to the system.
2. The manager selects the option of deleting employee/ project.
3. The manager selects the relevant employee/ project.
4. The system deletes employee/ project and update gaps in every project related to it.

תמונה שמכילה טקסט, צילום מסך, תרשים, קו

התיאור נוצר באופן אוטומטי

## 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.

**תמונה שמכילה טקסט, צילום מסך, תרשים, קו

התיאור נוצר באופן אוטומטי**

## 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.

**תמונה שמכילה טקסט, צילום מסך, תרשים, קו

התיאור נוצר באופן אוטומטי**

## 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.

תמונה שמכילה טקסט, צילום מסך, מקביל, תרשים

התיאור נוצר באופן אוטומטי

## 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.

**תמונה שמכילה טקסט, צילום מסך, תרשים, מקביל

התיאור נוצר באופן אוטומטי**

# Employee Use - Cases

## Insert Employee details:

### Description:

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

### Actor:

* The employee

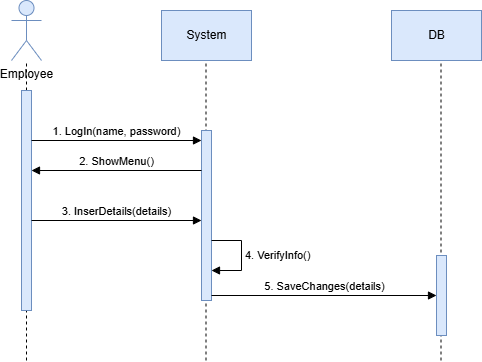
### Pre- conditions:

* 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.

## Update Employee details:

### Description:

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

### Actor:

* The employee

### Pre- conditions:

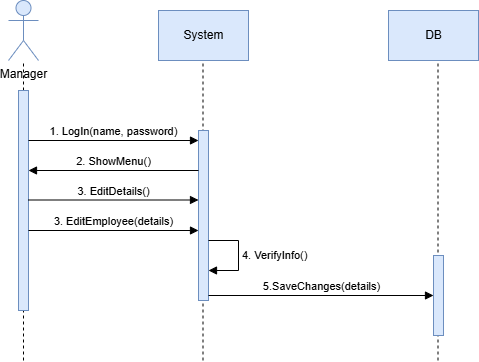
* The employee has access to the system, and is logged in.
* The employee is in the system.
* All the attributes are correct and logical (hours, age…)

### Post- conditions:

* The employee's details are updated.

### Main scenario:

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



## Open ticket:

### Description:

The employee opens a ticket for his absence.

### Actor:

* The employee.

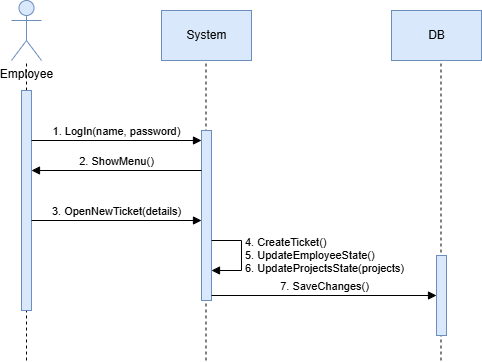
### Pre- conditions:

* 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. תמונה שמכילה טקסט, צילום מסך, תרשים, מקביל

   התיאור נוצר באופן אוטומטיThe system loads a page with all the employee's allocated projects and details.

# Big Management Use - Cases

## View Information per project and per employee:

### 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 information.

### Main scenario:

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

**תמונה שמכילה טקסט, צילום מסך, תרשים, קו

התיאור נוצר באופן אוטומטי**

Functional requirements

## 1. Employee and Project Management

* 1. The system shall allow team managers to add, edit, and delete
     1. employee records with attributes:
* Time zones for working.
* Programming proficiencies.
* Foreign language proficiencies.
* Level of English proficiency.
* Job Allocation Percentage: the proportion of time an employee is dedicated to specific tasks or projects.
* Technical expertise areas
  + **Development Domains:** Frontend Development | Backend Development | Full Stack Development
  + **API and Service Integration:** RESTful APIs | GraphQL | SOAP APIs | Microservices Architecture
  + **Database Management:** SQL Databases | NoSQL Databases | Data Warehousing | Data Query Languages
  + **Data Engineering and Analytics:** Extract, Transform, Load (ETL) Tools | Data Pipelines | Big Data Frameworks
  + **Cloud and DevOps:** Cloud Platforms | Containerization and Orchestration | Infrastructure as Code | CI/CD Pipelines
  + **Operating Systems and Platforms:** Windows and Linux Systems Administration | macOS Proficiency
* Years of experience.
* Company experience duration.
  + 1. project records with attributes:
* Project name and description.
* Final deadlines and estimated required hours.
* Technology stack requirements.
  + 1. project roles with required attributes and rank them by priority:
* Specific time zones for working.
* Foreign language proficiencies.
* Level of English proficiency.
* Job Allocation Percentage: the proportion of time the role requires.
* Technical expertise areas [refer to requirement 1.1.1]
* Minimum years of experience.
* Company experience duration.
  1. The system shall allow employees to input, edit, and delete details of:
* Time zones (for working).
* Foreign languages.
* Programming languages.
* Job Allocation Percentage: the proportion of time an employee is dedicated to specific tasks or projects.
* Upcoming vacations and constraints.
  1. The system shall display a dashboard showing:
* Each employee's current utilization.
* A summary of resource allocations for projects.

## 2. Issue Tracking

* 1. The system shall allow managers and employees to log absence-related issues, enabling managers and employees to log issues with required details:
* Detailed absence reasons [vacation | reserve duty | maternity leave | exams | long-term illness | grief | resignation | layoffs].
* Start and end dates of absence.

## 3. Gap Analysis and Resolution

* 1. The system analyze resource gaps resulting from reported absences and suggest:
* Suggestions for substitute employees based on required attributes of the role.
* Simulation scenarios showing potential impacts of changes on projects.

## 4. Reporting and Visualization

* 1. The system shall provide a graphical user interface (GUI) that allows the manager to:
* Adjust employee allocations interactively.
* Preview the effects of reassignment on:
  + Employee availability and workload.
  + Overall project status and timelines.

## 5. User Access

* 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
* Editing personal information
* Submitting tickets

## 6. Notifications and Alerts

* 1. The system shall notify employees and managers via email or in-app notifications about:
* Newly assigned roles and schedule updates.
* Pending absence issues and project deadlines.
* Resource shortages (employees overcommitted to multiple projects).

Non-Functional Requirements

## Performance:

* 1. Critical screens (Adding a Project, Opening a Ticket, and Handling a Ticket) must load within 2 seconds under a load of up to 10 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).

## Access Control:

* 1. Access to administrative features must be restricted based on user roles and permissions.

## 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 50 employees without performance loss.
  2. Adding new projects, teams, and requirements should not require significant changes to the system architecture.

Risk Assessment

1. Risk: Delays in meeting deadlines due to unplanned additions to the project scope or underestimating task complexity.
   * Mitigation: Define clear deliverables for each phase, and use Agile methodology for incremental progress and continuous feedback.
2. Risk: A complex and intricate system may lead to a decrease in the overall quality of the product.
   * Mitigation: Simplify the system design by prioritizing essential features, focusing on delivering a high-quality product in stages, and avoiding overcomplication
3. Risk: front-end may not be of sufficient quality due to the team's lack of expertise in this area
   * Mitigation: Quickly train the team through targeted courses and use established front-end frameworks to ensure best practices.
4. Risk: Dependency on third-party libraries or tools causing integration issues.
   * Mitigation: Choose reliable, well-documented libraries and maintain backups or alternatives.