ASSIST
Tech Challenge 2024

Team Finder App

Contents

Project Title – Team Finder

General Description (Business Context)

In large enterprises, but also in medium-size companies, the number of projects can increase exponentially, which leads to a real challenge when assembling project teams. To assemble a team, project managers may need to consider a series of skill sets across different departments and locations.

In this case, it becomes crucial to have access to the most up-to-date information about which people are available and what are their relevant skill sets. Furthermore, this process also needs to be compliant with the company structure and all its policies.

On top of that, the projects being developed have different stages, which might require a dynamic process of people allocation. In various stages of the project, the managers might need to increase or decrease the team, which leads to people becoming available or unavailable.

When depending on so many variables, it is very important to have an optimized process, which can ensure a fast and reliable result. Having multiple projects which finalize or begin various stages, it brings the need to take decisions fast.

Project Context

Imagine that you work for a company that specializes in providing project management solutions for medium to large enterprises, with complex and diverse project portfolios. As the company prospects for new products in this field, the sales team discovers a challenge that many of the customers are facing, assembling project teams with the right skill sets.

Considering the company business area, this looks like a great opportunity to build a new product, which would bring a lot of value to the company portfolio. As the decision has been made, a Product Owner is appointed with the task of performing some market research and collecting the product requirements.

After a few weeks of discussions with internal stakeholders and potential customers, the Product Owner was able to put together the list of requirements for an MVP (Minimum Viable Product).

After the requirements were approved by the company's directing board, your team was assembled and tasked with implementing the first version of this product.

Product Use Cases

Before going into the detailed requirements for the new product, it is important to understand how this platform will be used once it is available on the market.

For this, following use-cases have been described:

Use case – Find the best team

An outsourcing company has multiple departments with people specialized in various technologies. When a new project needs to be started, the managers need to assemble a team of experts with the right skills for the upcoming project. In this process, they need to understand which technologies will be used, which people are available for the new project and which of them have the necessary skills.

Fortunately, the company uses **Team Finder**, a platform which helps them document the technical profile (and skills) of each employee. Once the details of the new project are clear, the responsible manager uses the platform to assemble a suitable team for the project. For this, he/she needs to follow a few steps:

1. Create a new project.
2. Add project details (e.g.: used technologies, period, team size, needed roles, etc.).
3. Initiate a team find.
4. Review suggested candidates.
5. Add candidates to the new team.

Functional Requirements

The functional requirements were specified in an internal document, and they were grouped in 2 categories:

1. **Core/MVP Features** – features that are vital for this new system (they form the minimum viable product)
2. **Nice-to-have Features** – features that are a nice addition to the system (it will make the customers happy)

Core Features

[Auth-01] Sign Up as Organization Administrator

To use Team Finder platform, a user should first create an account. The only type of account publicly available is the *Organization Administrator*.

To create this account, a user should specify the following information:

- Name (name of the individual)
- E-mail Address
- Password
- Organization Name
- Headquarter Address (address of the main office).

When the sign-up is completed, there should be two entities created in the system:

- A new organization

- The new account attached to this organization

[Auth-02] Employee Sign-Up URL

Once an *Organization Administrator* logs in, the platform should provide a URL that could be used by other employees for signing up. This URL should contain information that automatically identifies the organization for which the employee accounts will be created.

When the URL is being accessed, it will lead to the employees sign up page.

[Auth-03] Sign Up as Employee

An employee of a specific organization can sign up on the platform by using the employee sign-up URL (see requirement [Auth-02]). To sign up, the employee needs to provide the following information:

- Name (name of the employee)
- E-mail Address
- Password

[Role-01] Access role assignments

The Team Finder platform is supposed to be used by multiple types of users within various organizations. Because of this reason, access to various information or features need to be restricted depending on roles. To keep the platform simple, following roles should be supported:

- **Employee**
    - It is the **default role** assigned to all users that create accounts using the employee sign-up URL.
    - Has the minimum access rights in the platform.
    - Can update his/her skills, in accordance with the skill assignment process.
    - Can view his/her projects and perform updates in accordance with the project update process.
- **Organization Admin**
    - Has all the rights of an employee, plus the ones corresponding to his/her role.
    - Creates the organization at the same time with his/her account (see requirement [Auth-01]).
    - Is able to assign any role to other users.
    - Is able to assign the role of Organization Admin to other users, which means that there could be multiple admins to an organization.
    - Can create departments
- **Department Manager**
    - Has all the rights of an employee, plus the ones corresponding to his/her role.
    - Is able to update the skill set for his department.
    - Is able to assign skills to his/her department members.
    - Is able to view the list of experts from his department or search for specific ones.

    - Can see the projects where his/her department members are assigned.
- **Project Manager**
    - Has all the rights of an employee, plus the ones corresponding to his/her role.
    - Is able to create and update projects.
    - Can search for experts with various skills (fit for his/her project).
    - Can assign/deallocate people to/from his/her project.

Considering the four roles presented above, following requirements have to be fulfilled:

1. An *Organization Admin* should be able to view all employees from his organization and assign the following roles to any of them, including to himself: *Organization Admin* (only to other users), *Department Manager, and Project Manager*.

Constraints

- A user could have multiple roles, in any combination. In the extreme case, a user could be *Organization Admin* + *Department Manager* + *Project Manager*. Having all these roles he/she would be able to access the features designed for all of them.

[Role-02] Custom team roles

As specified in the previous requirement, projects can only be created by users with the role of *Project Manager*. Also, when a manager assigns other users on the project they will probably be in the role of *Employee*.

On top of this, when working on a project, employees could have various roles in the team. Following list of roles could be an example: Team Lead, Technical Lead, Scrum Master, Frontend Developer, Backend Developer, QA Engineer, UX Designer, etc.

However, depending on how the organizations are structured, they may not use the same definition of roles. For that reason, Team Finder platform needs to allow the customization of **Team Roles**. In this regard, The *Organization Admin* should have access to a page where he can create and/or update a list of team roles for his entire organization.

Important Observations

- These roles are not used for granting access rights, so they should not be confused with the access role assignments defined in requirement [Role-01].
- They will just be assigned to employees in relation to the projects they are assigned to. Their purpose is to track which roles the employees had on those projects, so they are just informative.

[Department-01] Department Updates

Each *Organization Admin* should be able to create departments for his/her organization, by providing following information:

- Department Name

In addition, the organization admin should also be able to update or delete departments.

[Department-02] Assign Department Manager

An *Organization Admin* should be able to assign department managers for the departments created in his/her organization.

Constraints

- A user can be assigned as department manager of a specific department only if he/she has the *Department Manager* role.
- A department can have only one department manager.
- A user can be assigned as department manager only to one department.

[Department-03] Assign Department Members

When employees are creating accounts, they are not yet assigned to any department. In this case, a *Department Manager* should be able to view all unassigned employees and assign them to his/her department.

If an employee moves to a different department, the manager can also remove him from his department. Then it will be the responsibility of the new manager to add the employee in his department.

[Skill-01] Skill Updates

A *Department Manager* should be able to create, update or delete skills in the Team Finder platform. A skill should contain following information:

- **Skill Category**
    - The manager decides which categories to use. He/she should be able to create and reuse skill categories.
    - Examples: *Programming Language*, *Libraries*, *Frameworks, Software Engineering,* etc.
- **Skill Name**
    - A string representing the skill name
- **Description**
    - A description of the skill (text)
- **Author**
    - The name of the department manager who created this skill.
- **Departments**
    - A list of departments where this skill is used. Could be empty.
    - The manager can only add the department/s that he is a manager to. He/she is not able to add a department from another manager.

Constraints

- A *Department Manager* should be able to update only skills created by him/her
- A *Department Manager* can view skills created by other managers
- A *Department Manager* can add (link) skills created by other managers to his own department
- All skills created by department managers are visible only for the users from their organization.

[Skill-02] Skill Assignment

Any user from an organization (employees, admins, managers) should be able to assign himself/herself a set of skills. When assigning a skill, they should specify:

- **Skill**
- **Level**
  - One of: 1 – Learns, 2 – Knows, 3 – Does, 4 – Helps, 5 – Teaches
- **Experience**
  - One of: 0-6 months, 6-12 months, 1-2 years, 2-4 years, 4-7 years, >7 years

Besides assigning skills, a user should be able to see the list with all skills that they have.

[Project-01] Project Updates

A *Project Manager* should be able to create a project so that they can assign people to it. When creating a project, the following details need to be specified:

- **Project Name**
- **Project Period**
  - One of: *Fixed*, *Ongoing*
- **Start Date**
  - Date when this project officially starts/started
- **Deadline Date**
  - Only if the Project Period is *Fixed*
- **Project Status**
  - One of: *Not Started, Starting, In Progress, Closing, Closed*
  - During project creation, only the following statuses can be set: *Not Started* OR *Starting.* It should be forbidden to set one of the other states.
- **General Description**
  - A text with any relevant information about the project
- **Technology Stack**
  - A list of technologies/languages/frameworks/libraries used for development
- **Team Roles**
  - A list of roles needed in the team, as estimated by the project manager.
  - The roles that can be added here are the custom roles created in the organization by the *Organization Admin* (see requirement [Role-02]).
  - For each role, should be specified how many members.

After project creation, the *Project Manager* should be able to update any project data.

There should also be possible to delete a project under following conditions:

- The project **never had the status** *In Progress/Closing/Closed* AND
- A confirmation pop-up has been shown and the user confirmed the deletion

[Project-02] Team Finder

After the *Project Manager* opened a project, there should be a section/page where they can find employees who are available for the project.

When searching for employees, the *Project Manager* should be **able to configure availability criteria**:

- Include partially-available employees (people that are currently assigned to one or multiple projects, but less than 8 hours in total).
- Include employees from projects close to finish (people that are currently assigned to projects that have the deadline in maximum N weeks).
    - N can be specified by the *Project Manager* and should be limited between 2 and 6 weeks.
- Include unavailable employees (people that are currently assigned to some project, with a total of 8 hours)

The **result** should be a **list of employees that fit the following criteria**:

- They fit the availability criteria specified by the *Project Manager* (if none of the options above have been selected, only the fully available employees would be shown = people that are not assigned to any active project) AND
    - They have some skills who are matching the technologies mentioned by the project AND/OR
    - They worked in the past on a project that had similar technologies AND roles

[Project-03] Assignment Proposal

After getting a list of available people for the project (see requirement [Project-02]), the *Project Manager* can **propose** any member from the list to be assigned on the project.

When proposing the people for the project, the manager should specify the following information:

- **Work Hours**
    - How many hours per day should the employee be involved in the project
    - A number between 1 and the number of hours the employee has left (maximum 8)
        - the number of hours the employee has left = 8 minus the total hours assigned on other projects
- **Roles**
    - The roles that the employee will have in the team
    - Can be selected from the custom team roles (see requirement [Role-02])

- **Comments**
    - A text with any comments that the project manager wishes to add

[Project-04] Deallocation Proposal

Once an employee has been assigned to a project and validated by his/her *Department Manager*, they will officially be shown as part of the project team. However, if the project manager wants to remove people from the team, he can also propose this.

When proposing an employee deallocation, the manager has to specify the following information:

- **Deallocation Reason**
    - A text where the manager briefly explains the reason for removing that member from the team

[Project-05] Assignment/Deallocation Confirmation

All proposals for assignment/deallocation made by the project managers should be shown in a section/page to the *Department Managers* of each proposed employee.

The department managers should be able to view the information of each proposal and accept/reject them.

Depending on which proposal has been made and if it was accepted or rejected, the employee will become assigned or deallocated from that project.

[Project-06] View Project Team

The project team view can be accessed by the *Project Manager*, a *Department Manager* or the *Employees* that are assigned to that project (even after deallocation).

Because the team allocation process has multiple steps, the project team view should contain three sections:

1. **Proposed members**
    - Shows the employees that have been proposed for assignment (see requirement [Project-03])
2. **Active members**
    - Team members that have been assigned and confirmed by the department manager (see requirement [Project-05])
3. **Past members**
    - Team members that have been deallocated and confirmed by the department manager (see requirements [Project-04] and [Project-05]).

[Project-07] View Employee Projects

Employees (and *Project Managers*) should be able to view all projects that they have been assigned to. The projects should be shown in two sections:

1. **Current projects**
    - The active projects where the employee is currently assigned
2. **Past projects**
    - The projects where the employee was assigned in the past

For each project, following information needs to be displayed:

- **Project Name**
- **Roles**
    - The roles that the employee had on that project
- **Technology Stack**
    - The technology stack

[Project-08] View Department Projects

Projects are not directly assigned to departments, but employees from a specific department are assigned to projects. For this reason, a *Department Manager* should be able to view all projects where his/her department members are assigned.

The manager should also be able to filter by project status.

For each project, following information needs to be displayed:

- **Project Name**
- **Deadline Date**
- **Project Status**
- **Team Members**

Constraints

- A *Department Manager* should not be able to view a project if none of his/her department members are assigned to it.

[Project-09] View Project Details

When viewing a list of projects (see requirements [Project-07] and [Project-08]), a user should be able to open one project in order to see more details.

On this detailed section/page, following information should be present:

- **Project Name**
- **Project Period**
- **Start Date**
- **Deadline Date**

- **Project Status**
- **General Description**
- **Technology Stack**
- **Team Members**
    - active and past members, similar as requested in [Project-06]

Nice-to-have Features

[Skill-03] Skill Endorsement

When employees assign skills to their profile (see requirement [Skill-02]), they should also be able to specify a list of endorsements. An endorsement can be a **training**, a **course**, a **certification** or a **project** on which he worked in the past.

For each of the endorsements, they should specify:

- **Training/Course/Certification**
    - Title – title of the training
    - Description – a short summary of what he learned in that training (1-2 sentences)
- **Project**
    - Link to the project thy have worked on.

Constraints

- When adding a project as endorsement, the employee should only be able to select from the projects they have been assigned to. They should not be able to select projects that they have not been part of.

[Skill-04] Skill Validation by Department Manager

When employees assign skills (see [Skill-02]), they should appear in a dedicated section/page to the *Department Manager*. He/she can then view the skill assignment and validate it.

Skills that are not validated by *Department Manager* should not be shown on the employee profile, or taken into account in any way.

Each skill validation record from this section/page should contain the following information:

- **Employee Name**
- **Skill**
- **Level**
- **Experience**

Notes

- This requirement changes the behavior defined in the requirement [Skill-02]. If this requirement gets implemented, then after employees assign skills to their profile, they won't see them until they get validated by the *Department Manager*.

[Project-10] Assign Skills Requirements to Projects

Once a project has been created, the *Project Manager* or any other member of the project team, should be able to assign skills that are required for that project. This could be done in a dedicated section/page, which is editable by any project member.

A skill requirement should contain:

- **Skill**
  - A skill selected from the ones available in the organization
- **Minimum Level**
  - One of: 1 – Learns, 2 – Knows, 3 – Does, 4 – Helps, 5 – Teaches

Notes

- This requirement extends the functionality requested in [Project-01]

[Project-11] Skill Upgrade Proposal

Employees can have a section where they can check skill upgrade proposals based on their project participations. After every 3 months spent on a project, project team members can see skill upgrade proposals in this section, based on their role from the project and project skill requirements (see [Project-10]). These proposals are automatically inferred by the platform.

For every proposal shown on this list, there should be an option for the employee to add that skill directly to his profile, by specifying **level** and **experience** (similar as in requirement [Skill-02]).

[AI-01] Find Experts using a Chat-GPT Service

This requirement extends [Project-02] as follows:

- In the "Team Finder" section/page a new field is added, called "Additional Context". There, the *Project Manager* can add additional details (context) about the project, by using natural language.
- In the additional context field, the manager can include any details about the type of experts that he would like to get as a result, what skills/technologies they should know and so on.
- When the manager presses the "Find" button, these information should be sent to the OpenAI Chat-GPT service, along with the entire context from the database (employees, their skills, projects information, employee assignments on projects, etc.).
- The response from Chat-GPT service should be structured as JSON, so that it can be easily mapped to a list of employees.

- The result from Chat-GPT (in JSON format) should be processed and displayed in the same form as when searching experts with the first implementation (the one from requirement [Project-02]). However, the answer should also be logged in brute form, using a console log on UI. This will be used for debugging.

Constraints

- The request to OpenAI Chat-GPT service should be performed from the backend service. The backend assembles the request, sends it to Chat-GPT service, processes the response and returns the list of proposals to the FrontEnd.
- When sending the request to Chat-GPT, it should specify that the response should be in JSON format, not plain text.

[Notification-01] Notifications

For all users of this platform, there should be a section dedicated to Notifications. Also, either on the button/tab that leads to that section or in another visible spot, it should be displayed the number of unread notifications, so that users can easily see if they have new notifications.

Depending on the overall activity from the platform, following notifications should be implemented:

- *Department Manager* gets notified if one of his department members has been proposed for assignment on a project (see requirement [Project-03]).
- *Department Manager* gets notified if one of his department members has been proposed for deallocation from a project (see requirement [Project-04]).
- *Department Manager* gets notified if one of his department members has assigned a skill, which he/she needs to validate (see requirement [Skill-04]).
- An employee gets notified if he got a “skill upgrade proposal”, due to his involvement in a project (see requirement [Project-11]).

[Stats-01] Skill Statistics

For *Department Managers*, there should be a possibility to see all the skills assigned to their department and various statistics for each skill. This could be a dedicated section/page.

Following statistics should be available for each skill from the list:

- Total count of people having that skill
- Count of people having that skill at level 1 (Learns)
- Count of people having that skill at level 2 (Knows)
- Count of people having that skill at level 3 (Does)
- Count of people having that skill at level 4 (Helps)
- Count of people having that skill at level 5 (Teaches)

For some bonus points, the level-based and/or total count statistics could be displayed as a pie-chart.

Non-functional Requirements

This type of project could have many non-functional requirements, but we reduced the list for the purpose of this contest. As such, we will ask the following requirements to be met:

[Architecture-01] Web application

The developed solution should be a **web application**. From this point of view it should contain minimum three components:

1. UI Web Application, developed with one of the modern front-end frameworks (e.g. React, etc.)
2. One backend service
3. A database

[Infrastructure-01] Deployment in the Cloud

It is mandatory for all application components to be **deployed in a Cloud platform**.

Although there are multiple Cloud providers that could be used (Amazon, Google, Microsoft, etc.), ASSIST will offer support and materials on how to deploy applications on **Microsoft Azure**.

However, if a team decides that they don't need support on this aspect, they could use another provider as well.

[Management-01] Project Management

During the development phase, the team should use the **Project Management board provided by ASSIST**, on Azure DevOps platform: https://dev.azure.com/assist-tech-challenge

On this board, the team should organize their work in various Epics and Tasks. Each task should contain a description for what is being developed and it will be assigned to the team member doing the work. Along the way the assignee should update the task to the correct state, add necessary comments and record the time spent on the task.

The total time spent on a task should be logged in the corresponding section dedicated to the time tracking.

ASSIST will provide access to a dedicated board for each team and various materials/instructions so that the teams learn fast how to use it.

Recommended Azure Boards Tutorials:

- https://learn.microsoft.com/en-us/azure/devops/organizations/security/get-started-stakeholder
- https://learn.microsoft.com/en-us/azure/devops/boards/work-items/about-work-items
- https://learn.microsoft.com/en-us/azure/devops/boards/backlogs/backlogs-overview
- https://learn.microsoft.com/en-us/azure/devops/boards/boards/kanban-overview

Technology Stack

For the implementation of the application **ASSIST recommends following technologies**:

4. **Front-end**:
    - **Web UI App:**
        - Javascript/Typescript (React)
            - https://reactjs.org/tutorial/tutorial.html
            - https://websitebeaver.com/deploy-create-react-app-to-azure-app-services
        - Javascript (Angular)
            - https://angular.io/tutorial
            - https://henriquesd.medium.com/deploying-an-angular-application-in-azure-9f89edfe2b9c
    - **Mobile App (optional):**
        - Java/Kotlin (Android)
        - Objective-C/Swift (IOS)
        - Dart/Flutter
5. **Back-end:**
    - **Database:**
        - Azure Cosmos DB (NoSQL database)
            - https://learn.microsoft.com/en-us/azure/cosmos-db/nosql/quickstart-portal
            - https://learn.microsoft.com/en-us/azure/cosmos-db/account-databases-containers-items
            - https://learn.microsoft.com/en-us/azure/cosmos-db/nosql/how-to-model-partition-example
            - https://learn.microsoft.com/en-us/azure/cosmos-db/
            - Cosmos DB Clients:
            - C#: https://learn.microsoft.com/en-us/azure/cosmos-db/nosql/quickstart-dotnet?tabs=azure-portal%2Cwindows%2Cpasswordless%2Csign-in-azure-cli
            - Java: https://learn.microsoft.com/en-us/azure/cosmos-db/nosql/quickstart-java?tabs=passwordlesssync%2Csign-in-azure-cli
            - Python: https://learn.microsoft.com/en-us/azure/cosmos-db/nosql/quickstart-python?tabs=azure-portal%2Cpasswordless%2Cwindows%2Csign-in-azure-cli%2Csync
            - Node.js: https://learn.microsoft.com/en-us/azure/cosmos-db/nosql/quickstart-nodejs?tabs=azure-portal%2Cpasswordless%2Cwindows%2Csign-in-azure-cli
        - Azure SQL (relational database)
            - https://learn.microsoft.com/en-us/azure/azure-sql/database/
    - **Service:**
        - C# (ASP .NET Core)

        - https://learn.microsoft.com/en-us/aspnet/core/tutorials/min-web-api?view=aspnetcore-7.0&tabs=visual-studio
        - https://learn.microsoft.com/en-us/azure/app-service/quickstart-dotnetcore?tabs=net60&pivots=development-environment-azure-portal
    - Java (Spring Boot)
        - https://spring.io/
        - https://learn.microsoft.com/en-us/azure/app-service/quickstart-java?tabs=javase&pivots=platform-windows-development-environment-azure-portal
    - Python (FastAPI)
        - https://fastapi.tiangolo.com/
        - https://learn.microsoft.com/en-us/azure/app-service/quickstart-python?tabs=flask%2Cwindows%2Cazure-portal%2Cvscode-deploy%2Cdeploy-instructions-azportal%2Cterminal-bash%2Cdeploy-instructions-zip-azcli
    - Node.js (Express)
        - https://expressjs.com/
        - https://learn.microsoft.com/en-us/azure/app-service/quickstart-nodejs?tabs=windows&pivots=development-environment-azure-portal

Although we are recommending using the technologies from the list above, if a team would prefer to use different technologies for any of the mentioned components (Front-end/Back-end/Database) it is free to do so.

However, in that case ASSIST may offer limited support for those technologies if we don’t have available expertise. Nevertheless, if a team considers trying other technologies we strongly recommend to contact us via the team Slack channel to discuss their options.

There could be cases where we actually have the expertise and we could still offer support, so the teams should not discard their options just because of the list above. E.g., if a team would like to use **Ruby on Rails** for the backend, we have this kind of expertise in ASSIST and may be perfectly fine to use it.

UX Design

This is not a mandatory requirement, but if the team manages to create some wireframes for the UI, this will be rewarded with some extra points.

For this purpose, we recommend using the following tool: https://www.figma.com/

Learning resources: https://www.figma.com/resource-library/design-basics/

Notes:

- Figma has student licenses. If you create an account with your student e-mail address you could access the professional version, which has no limitations. See the following page: https://www.figma.com/education/higher-education/
- To obtain the maximum number of points, the following design activities should be followed:
    - User flows
    - Desk/Secondary research
    - Wireframes
    - Visual Design
    - Prototype

Quality Assurance

This is not a mandatory requirement, but if the team manages to create QA documentation, this will be rewarded with some **extra points**.

For the management of manual testing, the following platform should be used: https://www.testiny.io/

The platform allows creation of free accounts. On this platform, the QA can create following artifacts:

- Test cases
- Test plans
- Test runs

Learning resources:

- For a quick introduction, the following tutorial should prove useful: https://www.testiny.io/docs/getting-started/testiny-tutorial/
- For learning more about Testiny, we recommend the official documentation: https://www.testiny.io/docs/
- For a better understanding of the Software Testing field, we strongly recommend the following tutorial: https://www.guru99.com/software-testing.html

Test Automation

This is not a mandatory requirement, but if a team member is passionate about Test Automation we strongly encourage to implement some automated tests on this project. The effort will surely be rewarded with some **extra points** and they may win a place to one of our DevQA internship programs.

And to convince you how interested we are in this area, we have prepared a list of resources from where you can learn (see ANNEX 2 - Learning Resources Dev QA.docx).

Deliverables

During the competition, the following artifacts need to be delivered in various stages:

	Deliverable	Deadline
1	**Software Architecture Document**	03 March 2024, 23:59
2	**Initial Backlog**	
3	**Access to Git Repositories**	
4	**Task Management**	20 March 2024, 23:59
5	**Manual Test Management**	18 March 2023, 23:59
6	**Bug Tickets**	
7	**Automated Tests**	
8	**UX Design Project**	17 March 2023, 23:59
9	**Public Access to Application**	
10	**Acceptance Test**	19 March 2024
11	**Final Presentation**	21-22 March 2023

For **Project Management boards** and **pipelines**, ASSIST will create a project for each team in the following Azure DevOps Organization: **https://dev.azure.com/assist-tech-challenge**

Important note: all team members will receive access to this project.

1. Software Architecture Document

Deadline: 03 March 2024, 23:59

A document which describes the technical solution. See ANNEX 1 – Software Architecture Template.docx

The document, together with any additional resources (e.g. diagrams) needs to be uploaded in a Cloud Drive (e.g. Google Drive) and a public link will be sent to ASSIST.

ASSIST will provide an **online form** which will need to be submitted by the team leader with the link to this deliverable.

Also the link should remain valid until the contest is over.

2. Initial Backlog

Deadline: 03 March 2024, 23:59

Before starting the development phase, the team should establish a set of tasks (and subtasks) that will be carried out by the team members. This doesn't need to be a complete list of tasks for the entire project, but at least for the first milestone.

Example of tasks that could be created in this phase:

- Task for Creation/Writing of this document

- Task for design of the High-Level Architecture diagram
- Task for design of the Database model diagram
- Task for design of the UI Wireframes
- Task for creation of code repositories
- Task for creation of Azure Resources (e.g. Cosmos DB account, App Services, Form Recognizer service, etc.)
- Task for development and deployment of Hello World UI App
- Task for development and deployment of Hello World Backend App
- Tasks for implementation of the first 1-2 features

As you see, the first week will be quite busy in setting up all the projects, so you can get ready to start development. It is a good time to test every technology you will work with and implement a small test in it. Also, it is a good thing to integrate all the pieces together in this week, so that you can focus on development in the next ones.

3. Access to Git repositories

Deadline: 03 March 2024, 23:59

All source code should be committed to a public repository (Github), where the jury members have access to. At first, all teams should provide access to all repos for the following e-mail address: marian.pinzariu@assist.ro

In addition, ASSIST will provide an **online form** which will need to be submitted by the team leader with data about all repos. At a later stage, ASSIST will provide a list of e-mails for the rest of jury members that will need access to the repos.

All these information will be **communicated via Slack**.

Git tutorial: https://www.youtube.com/watch?v=1nQzh2D1YtI

The ASSIST team mentor will also offer support in using Git, so please reach him via Slack to call for help.

4. Task Management

Deadline: 20 March 2024, 23:59

During the contest, it is expected that the team members are creating and updating tasks on the Azure DevOps board, as well as recording their time spent on each task. It is **highly recommended to create tasks for all the activities performed in this contest**.

See the guideline provided by ASSIST on how to use Azure DevOps platform.

5. Manual Test Management

Deadline: 18 March 2023, 23:59

For the management of manual testing, the following platform should be used: https://www.testiny.io/

For more information and learning materials, check the **Quality Assurance** chapter above.

On this platform, the QA should create following artifacts:

- Test cases
- One test plan
- At least one test run

For the evaluation of this deliverable, ASSIST will assign a QA Engineer, who will request access to the Testiny project. The engineer will contact the team via Slack, close to the deadline date.

6. Bug Tickets

Deadline: 18 March 2023, 23:59

As the QA team members will test the application developed by the team, they should report bugs as "*Issue*" tickets in the Azure DevOps project (board).

After the deadline, an ASSIST QA Engineer will evaluate this deliverable by reviewing the bug tickets reported on Azure DevOps.

7. Automated Tests

Deadline: 18 March 2023, 23:59

If the QA team members decide to write automated tests, they should provide the following deliverables to ASSIST:

- Automated tests source code.
- A test-run report (only if the framework outputs a report after running the tests).
- A short video recording with the demo of a test run.

All these files should be uploaded in a Cloud Drive (e.g. Google Drive) and a public link should be provided to ASSIST.

ASSIST will provide an **online form** which will need to be submitted by the team leader with the link to this deliverable.

Also the link should remain valid until the contest is over.

8. UX Design Project

Deadline: 17 March 2023, 23:59

If the team created a design for the user interface, it can send it in one of two ways:

1. Files exported from the design tool (or screenshots)
    - Upload files in a Cloud Drive (e.g. Google Drive) and send a public link to the ASSIST Design Expert.

2. If an online tool has been used, send the link to the design and provide access for an ASSIST Design Expert.

The ASSIST Design Expert will contact the team via Slack, close to the deadline, to ask for this deliverable.

9. Public Access to Application

Deadline: 17 March 2023, 23:59

For the testing phase, ASSIST will require access to the running applications.

Since it is mandatory that all apps are deployed in Azure, we expect to be able to access the applications via public URLs.

If the team developed some native mobile applications (Android/iOS), they should travel to ASSIST Headquarters during the testing session, so they present their app on a mobile device.

ASSIST will provide a simple form where team leaders can submit all the public URLs before the deadline.

10. Acceptance Test

Deadline: 19 March 2024

Testing of developed apps (+ inspection of automated tests, if needed) will be performed by an ASSIST QA engineer, in collaboration with the team via online meeting.

- In case of mobile devices the team can send one or more members to the company office in order to perform the testing together with the assigned QA engineer.

Testing will take place on 19th of March, and it will be scheduled in a fixed time slot. ASSIST will communicate with each team on Slack to establish the testing time and inform the team about which QA engineer will be testing their app.

11. Final Presentation

Deadline: 21-22 March 2023

At the end of week 4, the team should present their results to the jury.

ASSIST will provide a draft presentation template, which teams can use to inspire for preparing, as well as details regarding the time and place of the presentations.

All these information will be communicated via Slack.

Evaluation Criteria

Category	Criteria	Max Points	TOTAL
Implementation Plan	High Level Architecture Diagram	10	45
	Database Model Diagram	10	
	List of Technologies	10	
	Initial Backlog in Azure DevOps	15	
Functional (Core Features)	[Auth-01] Sign Up as Organization Administrator	4	250
	[Auth-02] Employee Sign-Up URL	2	
	[Auth-03] Sign Up as Employee	2	
	[Role-01] Access role assignments	8	
	[Role-02] Custom team roles	4	
	[Department-01] Department Updates	4	
	[Department-02] Assign Department Manager	4	
	[Department-03] Assign Department Members	8	
	[Skill-01] Skill Updates	15	
	[Skill-02] Skill Assignment	11	
	[Project-01] Project Updates	12	
	[Project-02] Team Finder	18	
	[Project-03] Assignment Proposal	8	
	[Project-04] Deallocation Proposal	8	
	[Project-05] Assignment/Deallocation Confirmation	8	
	Project-06] View Project Team	9	
	[Project-07] View Employee Projects	8	
	[Project-08] View Department Projects	7	
	[Project-08] View Department Projects	6	
Functional (Nice to have Features)	[Skill-03] Skill Endorsement	14	
	[Skill-04] Skill Validation by Department Manager	8	
	[Project-10] Assign Skills Requirements to Projects	6	
	[Project-11] Skill Upgrade Proposal	14	

	[AI-01] Find Experts using a Chat-GPT Service	24	
	[Notification-01] Notifications	18	
	[Stats-01] Skill Statistics	20	
Project Management	Tasks documentation in Azure DevOps Board	15	**30**
	Time logging on every task	15	
Backend Development Quality	Quality of Git commits (frequency, etc.)	4	**25**
	Use of multiple logic layers (e.g.: controller, service, data access)	6	
	Code quality (function/variable names)	4	
	Code quality (implementation clarity/coherence)	4	
	Code quality (usage of data structures – collections, lists, etc.)	3	
	API quality (endpoint naming, parameter naming/types, etc.)	4	
Front-End Development Quality	Quality of Git commits (frequency, etc.)	4	**25**
	Creation of re-usable components (modularization)	13	
	Code quality (function/variable names)	4	
	Code quality (implementation clarity/coherence)	4	
QA	UI is User Friendly	6	**50**
	Low number of severe bugs found during testing	4	
	Quality of written test cases (details, used techniques, etc)	8	
	Quality of the Bug Report	8	
	Inputs validation (UI fields, etc.)	3	
	General Performance (Responsiveness)	3	
	Automation Tests	18	
UX Design Project	User flows (quality, details, etc.)	6	**35**
	Desk/Secondary research	6	
	Wireframes (quality, details, etc.)	6	

	Visual Design (quality, details, etc.)	9	
	Prototype	8	
Other	Quality of the final presentation	20	**40**
	Impression of the Jury	20	
TOTAL			**500**