Application definition

Objective:

Our project intends to help students manage their time and keep track of their future activities. For this purpose, we will develop a tool that allows its users to add and remove objectives. It will also have several options to keep track of them, like constant notifications and progress updates.

Users:

Primary: Students of 1st semester of UADY studying Software Engineering.
Secondary: Students of higher semesters of UADY studying Software Engineering.
Potential: All students at University level that struggle with time management.

There is a distinction between this profiles: The primary users are the ones in which we detect needs that should be solved and everything is intended to help them and they also provide the requirements; the secondary users are the ones who can take advantage of our project, that we know somethings about them but are not actively seeking to solve necessities as we don´t ask for requirements; and for the potential users we describe all the users that no matter their background, if they need our solutions, they can use them and also are a broader sample of people what our primary users show but on a bigger social scale, they also represents a bigger market if we intended to expand the project or develop requirements with more complexity.

Customers:

We don't have a specific client as no one is paying for the development and stakeholders are pending to be actively involved in the project like Scrum strongly suggest, so we decided to consider our customers as the same of our primary users because we have access to a current pool of students in first semesters to guide our vision, although for timing they won't be the ones using it, they can provide the depth needed to the project definition.

Innovation:

We will be the first product to propose a new way in which students can improve their organization by tracking their activities step by step and having reminders that will suggest new dates for continuing their activities and show their advance, so they not only finish on time but also have tools to retrieve time missed, the former might be provided by *reminder apps* but a combination with the latter, something that can help you “go back” in time, haven't been done. We noticed on a survey that a major problem is time management and as existing organization solutions are not working, we take part by proposing a product to give these methods a boost and help students move forward on their academic commitments.

Requirements or user stories

Distinguish which will be used:

Requirements, both functional and non-functional, are essential for the development process of software. They concisely describe, in a way that our client can easily comprehend, what our project should do and what its constraints are. That is why we have preferred them over user stories in our project, at least in this stage of the process.

Functional requirements and not functional requirements:

Functional:

- On the first use, the system shall provide a short introductory tutorial that teaches users its main features: add, edit and remove tasks, update progress, and change notification frequency.
- The user shall be able to add, edit or remove a task from their tasks list using a simple button.
- The user shall be able to set a due date when creating a task and also be able to modify it [due date] via the 'update task' button.
- The user shall be able to see a list of all of their current active tasks in an organized manner. It is very important to display due date and progress status.
- The system shall display the current progress of each task and allow the user to update said progress with a button.
- When the user has active tasks, the system shall keep pushing notifications until they complete them.
- The notifications shall use a language that captures the user's attention to prevent them from missing the assignment. For example: "Hey! You have a task due on [date]"; or when nearing a due date: "Important! You have an unfinished task due by tomorrow".
- The user shall be able to set the notification frequency of each task. The minimum shall be 1 notification per week. We think that allowing less than that will defeat the purpose of our project.
- Each notification shall give the user a direct way to update its task, preferably by adding a button directly on the notification.
- Each task shall allow users to append 'update notes' to them in order to keep better progress.
- Periodically, the system shall write a short sentence encouraging the user to work on finishing their tasks.

Non-functional (constraints):

- The system shall respond to user input within 1 second. According to a study (https://www.nngroup.com/articles/response-times-3-important-limits/), 1 second is the limit of the user's flow of thought, any more than that and they might get distracted.
- Our tool shall be accessible on mobile phones running Android 5.0+ or iOS 10+. This encapsulates more than 85% of smartphone users in the world (www.statista.com, developer.apple.com).
- Our tool shall be programmed using JavaScript, TypeScript or Python. We have noticed that these are the most popular programming languages for writing bots and webapps.
- Developers shall only use VSCode to program the software. VSCode is by far the most popular code editor and is well supported by our chosen programming languages.
- Our system shall not save any identifiable information from the users. We want to avoid all of the hassles that come with storing this kind of information.

- Our server shall run entirely on Linux. Most cloud providers don't require additional licenses in order to host apps using Linux, thus lowering the expenses.
- Our system should use a relational database. These are very well documented and are easier to implement.

Prioritization methodology:

For this project, we decided to implement the MoSCoW methodology. We believe it is perfect for the prioritization of requirements of a small scale team like ours. This technique is easy to understand, it assigns each requirement one of the following prioritization categories:

- Must have: Critical to the delivery of the current Sprint. If '*must have'* requirements are missing, the project delivery should be considered a failure.
- Should have: Important but not necessary for the increment delivery.
- Could have: Desirable requirements that could improve the user experience.
- Won't have: Lowest-payback requirements that have been agreed not to be planned for the time being. These requirements are either dropped or changed into more fitting ones.

To assign a category to each requirement, we evaluated them as a team and wrote-down our observations. Typically, you would want the client to have an input in this decision but we don't currently have a client. Either way, this is our classification:

Requirements	Observations	Classifications
Provide a short tutorial.	Should be easy to implement. It is essential for our users.	M
Add, edit and remove tasks.	Should be easy to implement. It is necessary for our project's functionality.	M
Be able to set and update due dates.	It is important and shouldn't be complicated, but we are not really sure. We have future intentions to change it to 'M'.	S
See a list of all current tasks.	Should be easy to implement. It is necessary for our project's functionality.	M
Display and update progress of each task.	Slightly more complicated to implement, but it is necessary for our project's functionality.	M
Push notifications.	We believe this is the hardest requirement to implement. To not stall the process we have decided to classify it as "S" with future intentions to bump it to "M".	S
Add friendly text to notifications.	It is important but depends on the notifications. It should not be a priority right now.	S
Be able to set notification frequency.	If we succeed in implementing notifications, this requirement should be easy. It is somewhat important.	S
Notification shortcut to task.	It is not necessary but it would be nice to	C

	implement it in the future.	
‘Update notes’ on each task.	It should be slightly complicated, but we believe we can achieve it. Not very necessary.	C
Encouraging sentences.	Not really important, not really hard to implement.	C

Diagram:
The team counts with artifacts that specify the requirements/user stories, it includes exceptions to take into account:
In the previous section, we tried our best to explain in detail all of our project's functional requirements and the product or artifact each one of them expects. In essence, what we look forward to having from the requirements is the following:

- A list of the user's tasks with buttons to create, update and remove a task.
- A display of each task's current progress with a button that allows the user to update said progress.
- Notifications to be pushed to the user's devices, constantly reminding them to finish their tasks and their respective due dates.

Development process
It’s distinguish in a clear way that we use an agile framework:
Process description:
Process management:
Meetings evidence:
Quality assurance:

Teamwork
Repository:
We used a GitHub repository that contains general description of our product, documentation and the three principal increments that are being considered. The documentation mainly includes the documents templates and binnacles of the process.

Metric of individual contribution:
The general contribution is calculated according to a formula that considers the average between the activities completed in time divided by Total activities plus the attendance divided by the highest attendance. Considering there exists more than three delays it will apply a sanction of 5% multiplied by the number of delays.

Verification of the contribution %:
The verification is registered in a table composed of the total activities and how many were finished on time, times that the activity was delayed and the number of team meetings and meetings with the mentor that each member of the team attended.

Subject proficiency

Generic and specific proficiencies in which the development process takes part are distinguished:

Generic proficiencies:

- Works with ICT in his/her professional interventions and in his/her private life in a suitable and responsible way.
- Works with others multi, inter and transdisciplinary environments in a cooperative way.
- Takes decisions in his/her professional and private practice in a responsible manner.

Specific proficiencies:

- Identifies the concepts linked to the phases of requirements, design, development, testing and maintenance, according to the recognized organisms of the discipline.
- Identifies human factors immersed in Software Development that contributes to the success of the Software project.
- Uses software engineering terminology properly in its professional interventions.

Demonstration of how the general proficiencies are met by specific activities of the development process:

- Works with ICT in his/her professional interventions and in his/her private life in a suitable and responsible way.
    - Throughout the whole project different tasks like “Work process” or “Meetings” require a different ICT tool, like meetings that are more accessible via TEAMS, Excel tracks better a checklist of contributions or Trello lets us visualize in a graphic way the pending activities. And for the future, more specific software like the ones used for modeling will also be tested to fit our needs and this abstraction thinking of deciding which one suits best is carried to our professional use of tools but also evolves our personal needs.

- Works with others in multi, inter and transdisciplinary environments in a cooperative way.
    - Some tasks like writing this document(from “Application definition” to “subject proficiency”) requires a different depth of knowledge, some team members are more skillful in english than others and we take advantage of that; in this way, tasks like creating a repository, the project vision, ways to organize and track progress and so on, are led by the ones with more experience that share their insights and helps the overall achievement of new concepts.
- Takes decisions in his/her professional and private practice in a responsible manner.
    - At the beginning of the increment each member makes a commitment based on their time and abilities (“Work process”), so being self aware takes great

relevance to make a responsible decision that affects your work and others. As joined tasks develop, the sub-teams are able to negotiate what is in the best interest of the daily scrum(e.g. in “Subject proficiency” Teodoro works on the repository management and Fernan on quality assurance) and in this sense, commiting to specific actions helps you develop a better decision taking mind.

Demonstration of how the specific proficiencies are met by specific activities of the development process:

- Identifies the concepts linked to the phases of requirements, design, development, testing and maintenance, according to the recognized organisms of the discipline.
    - All the phases(based on SWEBOK by IEEE) were considered in the creation of the timeline and have a determinate place in the organization of the documentation. We also considered making generic templates that can be filled according to the needments of the later phases. Activities like “Timeline” and “Scrum organization” are an example.
- Identifies human factors immersed in Software Development that contributes to the success of the Software project.
    - We considered dividing most of the activities and hash out the possible contribution of each member of the team depending on its abilities. We also prioritized the communication in the meetings and tried to be the most attentive as possible, we always maintained a respectful approach in the activities and valued the contributions of the different stakeholders. On the other hand in the early activity “Product definition” we asked users to give us their needs and for the next increment a new contact with them will occur.
- Uses software engineering terminology properly in its professional interventions.
    - Since our project was conceived and during “Product definition” and “Timeline” we adjust a holistic view of the development process. We take into account activities as: requirements eliciting, approving requirements, communication, scrum framework. In a broader perspective, we try to evaluate the feasibility of our project in question as “How are we going to develop this product?”, “Does there already exist another product that provides this service?”.