# **Project Charter Template**

## **Executive Summary**

*I like to include a high-level summary in this section, describing what the project is trying to accomplish, why the project has been initiated, what benefits are expected from the project, and so forth. This may change, of course, as the project pivots and the learn-measure-build cycle repeats, but having some context is very helpful to those doing the work. You never know what new and interesting ideas someone might come up with if they understand the big picture of where the project is trying to go and what it is trying to accomplish.*

## **Scope Statement**

*This section deals with more of the implementation details. So, for example in the “Executive Summary” section we documented the business reasons for the project, and now, in this area, we draw the boundaries around what sort of efforts we are going to go through to achieve the project’s goals.*

*Assume we are creating a new mobile application to interface with our core product, for example. Perhaps in the beginning we only want to create a minimum viable product (MVP) for one type of mobile phone until we validate that there is a demand justifying further development. As such, our scope statement might be something like this:*

*In Scope*

* *Create a new mobile interface to support customer requests through our application, but only support the Motorola Edge model running Android initially until customer validation has occurred.*

*We might also explicitly exclude items from the project’s scope too:*

*Out of Scope*

* *Support for offline functionality is out of scope for this project due to lack of customer interest and a possible lack of return on investment.*

*This clearly helps everyone stay on the same page and ensures that time, resources, and money aren’t spent on the wrong things. Sometimes you’ll find that documenting what is out of scope is more important than what is in scope!*

*And this is doubly so when dealing with outside vendors providing development work. In these instances, miscommunications about scope almost ALWAYS result in delays and additional costs. Best to head them off right at the start!*

## **Assumptions**

*This is a great place to call out and document those items that might become “gotchas” as the project progresses. I have found it is always easier to coordinate teams at the beginning of the project vs. halfway through, and identifying assumptions is a great tool to support this.*

*Let’s look at some actual examples:*

* *Company ABC will develop both the front end portions of the project, and the internal team will create the Python backend integration work.*
* *The internal and external teams will integrate as much as possible to maximize project visibility and touchpoints. As such, there will be a joint JIRA board, shared Slack channel, and Confluence documentation area created that will be fully accessible to both teams.*
* *The project's back and front end architectures can be worked on independently and concurrently. Work should begin on both portions, starting with the first sprint.*
* *The front end team can utilize mocks of the back end data model as provided by the internal team to develop against initially. These have already been created and are ready for use.*

*Each of these assumptions will clearly impact how the project progresses, and ensuring everyone understands them is critical to reducing misunderstandings and wasted effort.*

## **Deliverables**

*We want to be a little bit careful with this section. Since we are utilizing Agile Scrum, the deliverables may very well change over the course of the project. Perhaps after one of the sprint reviews or after a customer validation session, the Product team will modify the project’s roadmap/backlog contents.*

*I have found two methods to deal with this:*

*First method: Keep the deliverables section at a very high level, so that pivots and lessons learned don’t cause dissonance with the statements in this section.*

*Ex: Create a mobile application providing the customer access to their account details on a cellular platform.*

*Second method: Include more granular details, and don’t be shy about documenting changes to the deliverables in the updated versions of this document. I talked about this philosophy at the beginning of this article, for example.*

*Another consideration is what is expected when two organizations are working together (vendor and client, for example), and the final project hand-off occurs. This is a great place to list what will be needed for the project completion transition, so nothing falls through the cracks.*

### Internal Project Team

|  |  |
| --- | --- |
| **Deliverable** | **Notes** |
|  |  |
|  |  |
|  |  |

### External Project Team

|  |  |
| --- | --- |
| **Deliverable** | **Notes** |
|  |  |
|  |  |
|  |  |

## **Project Milestones**

## *We also want to be a bit careful with this section, too, so that we don’t delve off into the land of Waterfall project management. Enumerating a list of project milestones with due dates doesn't really support pivoting and applying lessons learned after each sprint cycle.*

## *However, I have found that the milestone area can be of great use as a set of event triggers for when to communicate with other departments (Sales, Marketing, and Support, in particular). I’m sure we’ve all worked on a project where development and testing wrapped up, the CEO is chomping at the bit to release, and sales and marketing find out to their horror that they now have a week to create all of the marketing materials, sales demo workflows, and the like because this project was never on their radar as nearing completion.*

## *The Project Milestone section; however, can provide those trigger points of when to communicate with the other teams and departments as key portions of the project complete, so we avoid situations such as the one described above. For example, a milestone for the user interface (UI) being completed could signal that the marketing team needs to access the project on the test server and begin taking screenshots for the brochures and email announcements. Or perhaps once the final testing milestone is reached the support team begins training on the new project’s functionality, so that when production rollout occurs they can resolve customer issues quickly.*

|  |  |  |
| --- | --- | --- |
| **Project Milestones and/or Phases** | **Team Members Involved** | **Status** |
|  |  |  |
|  |  |  |
|  |  |  |

## **Stakeholders**

*I usually write this section as a table of which Agile Product Owners are representing the project, any other internal team members that have a stake (ex: the head of financing for a billing system overhaul project), and anyone else who might be interested in project updates and news (ex: the customer representative who is managing the development contract).*

*I will also sometimes include team members from external parties if appropriate,*

### Internal Stakeholders

|  |  |  |
| --- | --- | --- |
| **Name** | **Department/Title** | **Contact Information** |
|  |  |  |
|  |  |  |
|  |  |  |

### External Stakeholders

|  |  |  |
| --- | --- | --- |
| **Name** | **Department/Title** | **Contact Information** |
|  |  |  |
|  |  |  |
|  |  |  |

## **Team Members**

## *I usually make two tables in this section: External team members and internal team members. This provides a “phonebook” of sorts so that it's easy for anyone on the project to find anyone else. I include all the technical people, the product owner(s), scrum master(s), etc.*

## *You can, of course, add additional tables and sections depending on the logical organization of the teams and/or companies involved in the project.*

### Internal Team Members

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Department** | **Contact Information** | **Project Role** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### External Team Members

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Department/Title** | **Contact Information** | **Project Role** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## **Procurements**

*Despite having a fancy heading title, this is mostly where I enumerate work that needs to be done to support the project.  For example, here are some example procurements from a recent project I managed:*

* *Create and provision git repositories*
* *Prepare development environment setup materials for the contracting team*
* *Create contracting team accounts for Confluence, Jira, and Slack*
* *...*

*Each of these items had a direct impact on our ability to quickly integrate the contracting team into the project, and having them in the PC makes remembering and tracking them easy.*

Known resources which must be procured to support the project:

|  |  |  |
| --- | --- | --- |
| **Description** | **Owner** | **Status** |
|  |  |  |
|  |  |  |
|  |  |  |

## **Project Resources**

*I dedicate this part of the document to documenting the locations (including links where possible) for all the project resources, such as wireframes, system documentation, look-and-feel guidelines, etc.*

*I often find having a separate table for each resource grouped by department or domain is helpful for organizational purposes. So, for example I often have two different tables for Product Management and Development resources.*

### Department ABC/Domain ABC

|  |  |  |
| --- | --- | --- |
| **Resource** | **Link(s)** | **Notes** |
|  |  |  |
|  |  |  |
|  |  |  |

### Department XYZ/Domain XYZ

|  |  |  |
| --- | --- | --- |
| **Resource** | **Link(s)** | **Notes** |
|  |  |  |
|  |  |  |
|  |  |  |

## **Communication Plan**

*Frankly, this is probably the most important part of the PC.  I document how we will communicate as a project team (i.e., email, slack, etc.), when meetings and other structured communication occur, whom to go to for general questions and the like for each team/department, and any conflict resolution procedures.  This section will likely be highly customized as per the needs of your project, team, external parties, and organization.*

### Communication Methods

Project communications will take place via one of these methods:

### Meeting Schedule

The following meetings will occur to support the project:

|  |  |  |  |
| --- | --- | --- | --- |
| **Meeting** | **Frequency** | **Purpose** | **Attendees** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### Out of Band Communications

For day-to-day ad hoc communications not falling into the meeting schedule above:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Area** | **Team Member(s)** | **Company** | **Title** | **Email** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

### Technical Conflict Resolution Process (Optional)

*Example:*

*Should the ABC and XYZ development teams disagree about the technical details, implementation and/or design, the following conflict resolution process will be followed:*

*<List whatever conflict resolution steps are appropriate here>*

*Once the decision has been made on how to proceed, utilize one or more of the steps above:*

* *The final decision will be documented and disseminated out the engineering teams along with any appropriate supporting documentation, narrative, scope change, etc.*
* *...*

## **Risks**

*This doesn’t need to be a full-blown PMP style risk discovery, documentation, and planning session.  I find with Agile projects a simple table of the risks along with the mitigation plan suffices.  You might also include who is responsible for monitoring and executing the mitigation actions if appropriate.*

|  |  |
| --- | --- |
| **Risk** | **Mitigation** |
|  |  |
|  |  |
|  |  |