CoSpace
Project Plan

1. Introduction

This document lays out the organization and planned processes and milestones, as well as the rough estimations of how much effort will be needed for the completion of the development project CoSpace.

2. Project Organization

2.1. Team

Our team "overengineers" will be a highly cohesive team, where everyone will be aware of, and have a say in any part of the project. We will seek high levels of coordination and cooperation, with constant communication. Nevertheless, we will have specialized roles in addition to software developer roles, where individual members will be responsible for the tasks that are assigned to their title. However, this will not be their clearly cut sole responsibility, as everyone will have a say at any stage in the project.

Below are the roles assigned to the team members. Following an internal discussion, these roles are assigned randomly, as members of the team have the same level of professional experience regarding any role. But we do know that normally this would not be the case.

Team Member	Software Developer	Project Manager	Software Analyst	Software Architect	Software Configuration Manager	Software Tester
Şamil Ateşoğlu	X	X				
Yusuf Keten	X		X			
Mert Demir	X			X		
Selim Şeker	X				X	
Çağatay Yiğit	X					X

2.2. Descriptions of the Roles

2.2.1. Software Developer

Software developers will be responsible for the actual construction of the software and its components. They will understand the requirements and write software that conforms to those requirements, implement tests.

They will also coordinate with software analysts to eradicate any questions about the functionality of the software they are building, with software architects to get a better understanding of the high level structure of the project to build maintainable software and to act in a consistent way that conforms a chosen standard or standards regarding software construction, with software configuration manager to keep track of changes to the software & other artifacts and to understand the organization of repositories, with software testers to assure quality of the software and to locate and fix problems arose during testing.

2.2.2. Project Manager

Primary responsibilities of the project manager include:

- Facilitating communication and coordination among team members
- Constructing and following a project plan, estimating effort and budget required to complete the project
- Identifying and triaging individual development tasks
- Identifying the risks involved and specifying how to mitigate them

2.2.3. Software Analyst

Primary responsibilities of the software analyst include:

- Understanding the needs, objectives and requirements of the product
- Specifying requirements at various levels of sophistication: Vision for the project, and specific technical requirements to be fulfilled
- Relaying requirements to team members, and making sure they understand what they are building
- Avoiding any misunderstanding between the receivers and builders of the software

2.2.4. Software Architect

Primary responsibilities of the software architect include:

- Designing an architecture for the solution that makes the software easier to maintain and change
- Specifying roles of the software components and interactions between them
- Keeping a documentation for the architecture to maintain a better understanding of the system among team members
- Enforcing conventions and standards regarding software construction

2.2.5. Software Configuration Manager

Primary responsibilities of the software configuration manager include:

- Identifying configuration items (artifacts) that will affect the end product; such as source code repositories, requirements specifications and other documents
- Facilitating versioning of configuration items to form commonly accepted baselines
- Controlling the changes of configuration items to ensure consistency between them
- Identifying change procedures, reviewing change requests, approving/rejecting them

2.2.6. Software Tester

Primary responsibilities of software tester include:

- Reviewing requirements and prepare testing scenarios
- Performing quality assurance by testing the use cases and determining their conformance to the requirements
- Analyzing the impacts of defects and reporting them
- Coordinating with team members for locating and fixing problems arose during testing

3. Development Process and Measurements

We will be following an iterative development process, Rational Unified Process. We will track project progress and tasks using GitHub kanban boards, milestones and issues. We will also make a preliminary estimation about the project size, as well as the effort to complete the project.

3.1. Rational Unified Process

The Rational Unified Process describes how to effectively deploy commercially proven approaches to software development for software development teams.[1] The software lifecycle is broken into cycles, each cycle working on a new generation of the product. The Rational Unified Process divides one development cycle in four consecutive phases, which are described in the following sections.

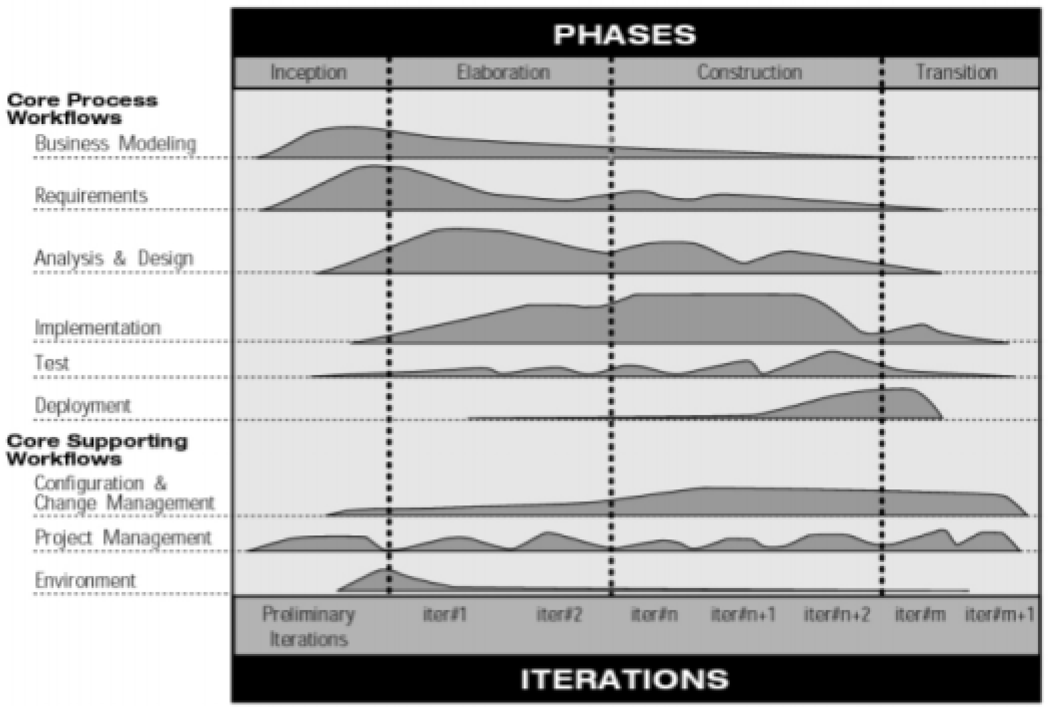

Figure 1. General overview of the process that will be followed

3.1.1. Inception Phase

During the inception phase, the needs for the product are understood and the scope of the project is defined. All external entities which are affected by the need and the solution proposed, are identified (stakeholder analysis). Core requirements and features of the product are identified and main constraints to be considered are described. The initial plan for the project is outlined. Risks that may hinder the project progress are also identified as well as how to mitigate them, and initial effort and budget estimations are carried out.

The outputs that we will produce during this phase will include:

- A vision document that outlines the needs for the product and main features to be implemented *(DEL1)*
- An initial project plan document that outlines the processes to be followed during project development *(DEL1)*

[1] Rational Unified Process: Best Practices for Software Development Teams

3.1.2. Elaboration Phase

During the elaboration phase, further specification of requirements, use case definitions, and technical constraints are identified. Essentially, high level objectives and features of the product are *elaborated* in detail for the implementation. As per any other phase, the outcomes of this phase are also crucial for common understanding between our team as well as other parties involved. As the requirements, use cases, and risks become specific and more clear, better estimations about the project cost and effort are carried out, all of which are important deciding factors for resource allocation. Once the software requirements will become clear enough, we will gradually start the implementation during this phase.

The outputs that we will produce during this phase will include:

- Use case definitions that detail how will the product be used *(DEL2)*
- Functional and non-functional requirements, specifying precisely what will the system do *(DEL2)*
- Test case definitions that detail which tests are going to be carried out to validate the system's conformance to use cases *(DEL2)*
- List of graphical user interface designs that show how user interface components are designed and laid out *(DEL2)*
- A software architectural notebook that technically defines the software architecture and the rationale behind certain software design decisions *(DEL3*)

3.1.3. Construction Phase

During the construction phase, software components and features will be integrated into the product. Each team member will take on the responsibility of identified tasks and work on them in parallel, continuously integrating new features into the product. We will document the design of the software as well as conventions we follow, which configuration management processes we defined, risks we faced and how we mitigate them.

The outputs that we will produce during this phase will include:

- A working prototype of a core use case
- A risk management report that documents which risks we actually faced and mitigated *(DEL3)*
- A configuration and change management report that documents our change management processes *(DEL3)*
- A software design document that details the design via UML diagrams of the actual software we are building *(DEL4)*
- A coding standards document that details common practices and conventions we will follow and are following *(DEL4)*
- Mostly complete source code of the system

3.1.4. Transition Phase

During the transition phase, the software will be further tested internally. After validation, it will be deployed and beta testing will be performed to validate the end-user experience.

The outputs that we will produce during this phase will include:

- A software test result report, that will include results of the tests performed *(DEL5)*
- A presentation that will introduce the final software *(DEL5)*
- Actual source code of the system *(DEL5)*

3.2. Measurements

Size and effort estimations are **especially ineffective** so early in the project development. But we will conduct them to get an idea about the magnitude of the project, using the function point methodology and early design model of COCOMO II.

Function Point Type	**Name**	**Complexity**
External Inputs	New Post Page	Complex
	Register Page	Average
	Fill Questionnaire Page	Complex
	Recommend New Sub-Club Page	Average
	Conduct Meeting or Event Page	Complex
	Ban User or Moderator	Average
	Prepare Questionnaire Page	Complex
	Set Moderator	Complex
	Create Club/Sub-Club Page	Average
External Outputs	Post View Page	Complex
	Home Page	Complex
	Club Feed Page	Complex
	Club Tree View	Complex
	Search Page	Complex
	Profile Page	Average
	Admin Page	Complex
	Moderator Page	Complex
	Event List View	Complex
External Queries	User Search	Average
	Post Search	Average
	Club Search	Average
	User Info Retrieval	Simple
	Post Info Retrieval	Simple
	Club Feed Retrieval	Complex
	Club Event Retrieval	Average
	Questionnaire Retrieval	Average

Internal Logical Files (Tables)	User Relation	Average
	Post Relation	Complex
	Club Relation	Complex
	Admin Relation	Average
	Moderator Relation	Average
	Event Relation	Average
	Questionnaire Relation	Average

Above are the very tentative, probable function points. At the *very best*, they are educated guesses. Based on above assumptions, we have the following:

Function Point Type	**Simple**	**Average**	**Complex**
External Inputs	0	4	5
External Outputs	0	1	8
External Queries	1	5	1
Internal Logical Files (Relations)	0	5	2

This gives us an unadjusted function point of 216. Using the complexity adjustment questionnaire[2] we get an adjustment factor of 1.05, which makes our adjusted function point count 227. Using the latest function point conversion table from QSM[3] for Java, a median value of 53 lines per FP is multiplied with 227 FP:
$53 \times 227 = 12031$ estimated lines of Java code.

Early design model for COCOMO II states that $Effort = A \times Size^{B} \times M$ where A is a predefined co-efficient, *Size* is expressed in Kilo Source Lines of Code (KSLOC), B and M are adjustment co-efficients based on some hyper parameters. If we take all of them nominal, B value would be 1.1, M would be 1. A is proposed by Boehm to be 2.94 based on analysing a large dataset of software projects.

$2.94 \times 12.031^{1.09} = 45.3$ person-months of effort,

which indicates that this is a relatively small project, but it might not be applicable to a contemporary web application with large number of libraries available and very well software-reuse opportunity.

[2] http://groups.umd.umich.edu/cis/course.des/cis375/projects/fp99/table.html
[3] https://www.qsm.com/resources/function-point-languages-table

4. Project Milestones and Objectives

Phase	Iteration	Primary objectives	Scheduled start and end dates	Estimated effort
Inception	I1	Objectives 1. Prepare a vision document that defines the common understanding of the need for the project and its high-level core features. *(DEL1)* 2. Prepare an initial plan document that outlines the processes that will be followed and deadlines to be considered *(DEL1)*	4.3.2021/14.3.2021	10 hours
Elaboration	E1	Objectives 1. Prepare detailed software requirements specification document that will include use cases, functional and non-functional requirements, test cases, graphical user interface sketches *(DEL2)* Potential Risks Involved 1. Incorrect understanding of requirements. Mitigation: Effective communication between team and customers	14.3.2021/28.3.2021	20 hours
Elaboration	E2	Objectives 1. Prepare an architectural notebook that defines the decisions and considerations that will shape the system design *(DEL3)*	26.3.2021/3.5.2021	16 hours
Construction	CN[4]	Objectives 1. Implement the features outlined in software specification document 2. Prepare a software design document using UML diagrams and other tools *(DEL4)*	14.3.2021/23.5.2021	-

[4] During the construction phase, we will divide the work into an arbitrary number of iterations, all of which will be determined upon regular meetings.

		3. Prepare a coding standards document regarding the common practices and conventions followed *(DEL4)* 4. Prepare risk management reports *(DEL3, DEL5)* 5. Prepare configuration and change management reports *(DEL3, DEL5)* Potential Risks Involved 1. Underestimation of skills required to develop a part of the system. Mitigation: Effective communication, cooperation between team members. 2. A member of the team has fallen ill. Mitigation: Reassignment of tasks as soon as possible during the ill period.		
Transition	T1	Objectives 1. Prepare a software test result report *(DEL5)* 2. Conduct beta testing Potential Risks Involved 1. Unexpected bugs are reported, and their solution requires significant code changes. Mitigation: Always follow common coding standards, and implement unit tests during development.	15.4.2021/26.5.2021	10 hours
Transition	T2	Objectives 1. Prepare a final presentation *(DEL5)* 2. Ship final product *(DEL5)*	20.4.2021/26.5.2021	16 hours

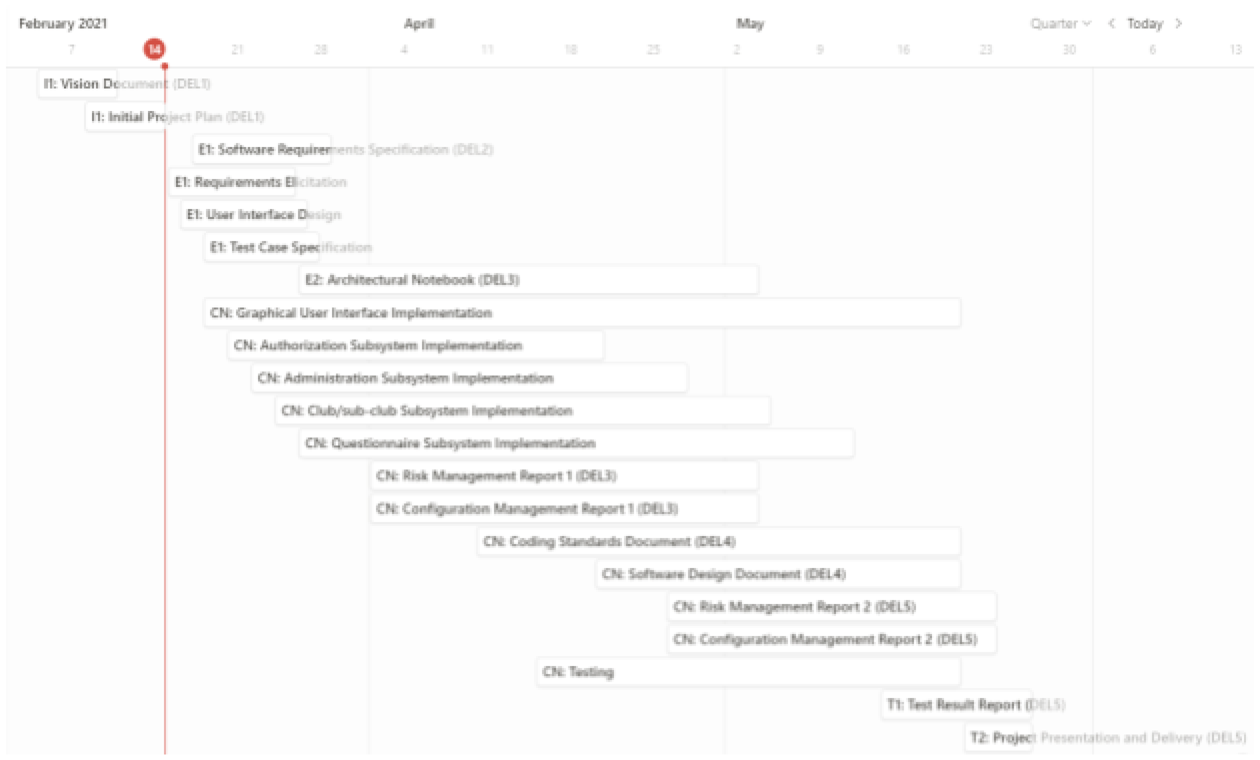

Figure 2. A Gantt chart demonstrating the estimated project schedule

5. Deployment

A suitable Continuous Integration/Deployment tool will be selected upon internal assessment and will be utilized to ship the product and deploy changes.

6. Lessons Learned

- To make the common understanding clear, effective and continuous communication is essential.
- Software effort estimation is hard and usually unreliable especially early in the development.