Delivery 1: System Requirements and Project Plan

1. Business Case

1.1. Team

Describe your team - team name, members.
Every team member has a role. Specify what exactly every member of the team will do, and what they are responsible for.
A team must have a leader.

1.2. Product or service

In subchapters describe your software product or service (referred to as “system” later in this document) that you will be building at a high level.

1.2.1. Purpose and mission

Describe why should the system be created? Who will benefit from it and how?

1.2.2. System description: user’s perspective

Overall description of the system from a user’s point of view (in free form). Explain who the users are, what they want to achieve by using the system, what are their habits and expectations, what alternatives they have, how many users are expected, would they be willing to pay for using the system (if applicable), etc.

1.2.3. System description: provider’s perspective

Overall description of the system from your (as provider of product or service) point of view (in free form). Explain what business your team represents, how will the system change the daily life of that business, how good the system will be, why is it needed and how it is better than alternatives, etc.

1.2.4. Interface prototypes

Provide prototypes of principal interfaces through which users (both people and other systems) will interact with the system. For example, depending on the type of system: screen views, list of web service endpoints and their descriptions, list of command line commands and their description.

1.2.5. Additional information (optional)

Any additional information that you would include for investors/sponsors to help them make a decision to fund your project.

2. System Requirements

2.1. Functional requirements

Describe the functions of your system. Should be specified in a use-case matrix.

2.2. Non-functional requirements

Describe the restrictions or qualitative requirements e.g. performance, security, reliability, usability.
This includes all the constraints that your system must meet. For example, “Web page should look well on a 5” mobile device”, “Tool must run on Windows 8 or later”, “The system must be translated in multiple languages”, and so on.

3. Project plan

3.1. Main risks

Identify the main risks that could occur while trying to build the system and meet the deadline.

3.2. Work breakdown

Every project can be broken down into smaller units of work. It can be functions or their groups (features), as well as steps to meet non-functional requirements. Describe what are the units of work and what is the reasoning behind it (e.g. why not some different breakdown?).

3.3. Schedule

Assign planned date to each work breakdown item, list any other important dates.

Delivery 2: System Design

4. System Design

While subchapters will describe some views of design, here your should provide a very brief description of the high-level design (architecture). It could include system metaphor, architectural style, or just the main highlights from the design.

4.1. System use cases

Describe the system and its context. Include all actors interacting with the system.
UML use case diagram must be used within this chapter.

4.2. Domain model

Model the classes within the domain, as well as their relationships using a UML class diagram. Keep in mind that the model should be rich, and not anemic. UML sequence diagram(s) should be used to explain the most complex interactions within the system.

4.3. Subsystems or main components

Identify the main components/subsystems and document them using a UML component diagram.

5. Technologies

5.1. Technology stack

Describe choices of programming languages, libraries, frameworks, and other technologies or tools with rationale of those choices.

5.2. Operational environment

Describe in what environments the system will run and why.

Delivery 3: Prototype and Technical Documentation

6. Development tools

In this chapter describe your development process and tools supporting it. Essentially a fellow developer should be able to read this part of the document and start working as part of your team.

6.1. Version control system

Describe what version control system do you use and how, including (but not limited to) whether it is distributed or centralized, what branching strategy do you use, etc.
Description should also include instructions on how to check out your code.

6.2. Code reviews

Describe how you perform code reviews. Who reviews what and when? Do you have a coding standard? Are reviewer approvals mandatory? Are all comments mandatory to address? Etc.
Description should include instructions on how to request a code review, and how to find pending and historical reviews.

6.3. Build and CI/CD pipeline

Describe how your project is built and how to do it from scratch on a local machine.
Then describe your CI/CD pipeline. When is it triggered? What happens at every step? What happens if a build fails? What happens if it succeeds? What tools do you use?
Description should include instructions on where to find a list of builds and how to trigger them.

6.4. Test automation

Describe how your system is tested. How many test cases and at which levels do you have? Which of them are automated? Which tools do you use for test automation?
Descriptions should include instructions on how to execute tests.

6.5. Issue tracking / Planning tools

Describe what tools do you use for tracking tasks and defects, planning releases, etc.
Description should include how to use those tools, e.g. what type of issues to use when creating a task, how to know which task to take next, etc.

7. Deployment

7.1. Deployment diagram

Describe how physical artifacts comprising your system are deployed on nodes using UML deployment diagram.
Description should include an overall of your whole infrastructure.

8. Additional information (optional)

Mention any other things you think are relevant or interesting.
For example, how do you monitor your system? Do you have any static code analysis tools? How do you ensure the security? Maybe you have some particularly smart solution/setup that works very well?
These are just some ideas, but anything can be included.

Additionally to the document outlined above for the 3rd delivery:

- Both Virgilijus and Gediminas must be granted access to your source code repository
- You must have and be able to demonstrate a working prototype of your system