Multimedia Services : Product Planning

Group BeNine
TI2316 Context Project

Bryan van Wijk (bryanvanwijk, 4363329)
Dorian de Koning (tcmdekoning, 4348737)
Ege de Bruin (kedebruin, 4400240)
Jochem Lugtenburg (jlugtenburg, 4370805)
Naomi de Ridder (nderidder, 4383109)

April 28, 2016

Supervisor: Dr. Cynthia Liem
Software Aspect TA: Valentine Mairet
Context Aspect TA: Alessio Bazzica

Delft University of Technology
Faculty of EEMCS

Contents

1 Introduction

This document contains information about the planning, and constraints involved with the project. The contents of the documents are divided in three main chapters. First, a high-level overview of the product backlog, and a general Roadmap will be given. Second, a more detailed product backlog is given. The final chapter contains the Definition of Done.

2 Product

This section describes the general form and content of the product backlog, as well as a Roadmap of progress during all stages of the project.

2.1 High-level product backlog

User stories are documented using the MoSCoW Method. The MoSCoW method is a widely used standard in the industry and involves prioritized requirements. The requirements are categorized in four categories, 'Must', 'Should', 'Could' and 'Would/Won't'. If a 'Must' is not included in the final deliverable, the project is not considered a success.

Must haves

Among must-haves, the team considers essential elements of the final product. These include camera operations, control and intelligent presets.

Should haves

Non-essential features improving user experience are marked as should-haves. These include editing camera presets during live performance.

Could haves

Features expanding the capabilities outside of the main scope of the product are considered could-haves. In the event of extra time, the team may consider to implement some of them.

Would/Won't haves

In the case of a feature which will definitely not be implemented, we mark it as a won't-have. This includes a script editor, since the client is already in possession of such a program.

2.2 Roadmap

Now, a general overview of each sprint will be given, formatted in a comprehensive list.

Initial Phase

The initial phase involves a visiting the client and designing the high-level architecture of the program.

Sprint 1

- Simple web interface and basic server infrastructure.

Sprint 2

- Implement basic functionality for presets.
- Create computer-vision preset plan.

Sprint 3

- Begin implementing computer-vision preset plan.
- Begin implementing intelligent presets.

Sprint 4

- Implement computer-vision preset plan.
- Preset creation interface.

Sprint 5

- User feedback on implementation.
- Expand preset creation interface.
- Implement director interface.

Sprint 6

- Expand director interface, improve integration between different interface.

Sprint 7

- Extension, used for user-feedback or additional features encountered in the process.

Sprint 8

- Finalize product, resolve bugs.
- Prepare product for release.

3 Product Backlog

This section describes all of the different user stories using the MoSCoW method. The MoSCoW method is a widely used standard in the industry and involves prioritized requirements. The requirements are categorized in four categories, 'Must', 'Should', 'Could' and 'Would/Won't'. If a 'Must' is not included in the final deliverable, the project is not considered a success.

3.1 User stories of features

Must have

As a user I want to

- be able to operate camera's using a web interface on my tablet computer.
- have an interface where I don't have to press multiple buttons to select a camera.
- configure custom presets, to select multiple cameras at once.
- store multiple presets per camera.
- load one of the presets.
- to be able to define intelligent presets, to reduce the amount of time involved in creating a good shot.

Should have

As a user I want to

- be able to set the current camera as preset.
- be able to change the presets.
- be able to see a preview of my presets.

Could have

Won't have

As a user I want to
As a user I don't want to

- create a (comprehensive) script editor.

3.2 User stories of defects

Must have

As a user I want to

- not lose my presets after the software crashes.
- be able to recover from a software crash quickly.
- be able to continue work on the other crashes if one of the tablets crashes.

Should have

As a user I want to

- be able to see what caused a software crash.

Could have

Won't have

3.3 User stories of technical improvements

Must have

As a user I want to

- be able to replace my cameras, requiring minimal changes in the software.

Should have

Could have

Won't have

3.4 User stories of know-how acquisition

As a user I want to

Must have

- be able to use the program without requiring a high amount of training.
- be able to consult a brief manual on how to operate the product.

Should have

Could have

Won't have

3.5 Initial release plan

4 Definition of Done

To conclude this document, a set of rules will be given which are used to judge if a certain task from the sprint plan can be marked as 'Done'. This set also applies to integration of branches into the main branch, and the review of deliverable documents.

4.1 Sprint Plan Tasks

A sprint plan task is considered 'Done' if it complies with the upcoming three rules. First, the work done should match with the description mentioned in the Sprint Plan document. Second, if a user story is involved, this user story must be fulfilled and properly tested. Finally, the majority of the team members should agree upon marking the task as 'Done'.

4.2 Features

To mark a feature as 'Done', a predefined procedure has to be followed. All features implemented, must be implemented in a branch which is not 'master'. The feature(s) implemented on this branch should be thoroughly tested, with a test coverage of at least 75% and must be following the code conventions dictated by the Checkstyle plugin in Maven. Errors of warnings generated by PMD and FindBugs should be eliminated if possible. Proper documentation should be added to ensure that people who are not involved in the project, or people who haven't been working on a feature for a long time can understand the decisions and methods used to write the code, so they may extend it. During integration, in the form of a GitHub pull request, the team reflects the rules above. If the majority of the team agrees, the feature branch may be integrated into the master branch, marking the feature as 'Done'.

4.3 Deliverable Documents

Deliverable Documents must comply with the points described in the 'Deliverables' section of the general context project guidelines document. Each document should also be written in clear, formal English language.

4.4 Sprint and Final Product

A Sprint is successful if all tasks, are marked 'Done'. If any problems occur, the team must reflect on them during retrospective and improve in the next iteration. A successful Final Project should be accepted by PolyCast, the client and must meet the quality aspects described in the previous subsections. The must-haves described in the Product Backlog are very important. If the team fails to include one of these cases the product might not function, or it may be unpleasant to use. To improve the user-experience, a reasonable segment of should-haves should also be implemented.

Glossary

MoSCoW Method is a widely used standard in the industry and involves prioritized requirements.. 4