Emil Karlsson & William Asp

Assignment 5 - Project

User Stories

Shared or overlapping

Login
As a user in the system I should be able to login with my given username and password
GUI: 10 min, logic: 40 min

Workflow 1 - Events

Create Client
As a SCS I should be able to create new clients
GUI: 20 min, logic: 30

Create Event Request
As a CS I should be able to create new event requests
GUI: 20 min, logic: 30

Review event (Financial)
As a financial manager I should be able to add information regarding financial details
GUI: 40 min, logic: 45 min

Review event (Administration)
As a reviewer I should be able to see the financial details in the event request
GUI: 40 min, logic: 45 min

Forward requests
As a reviewer I should be able to forward request to other managers for further review
GUI: 30 min, logic: 20 min

Workflow 2 - Tasks

View tasks
As a member of a subteam I should be able see my current tasks and information about it
GUI: 1.5 tim, logic: 30 min

Comment on tasks
As a member of a subteam I should be able add comments to a task, which PM/SM also can see
GUI: 30 min, logic: 40 min

Manage tasks
As a member of a subteam I should be able to mark my tasks as complete
GUI: 50 min, logic: 30 min

Create tasks
As a SM/PM I should be able to create tasks for a specific subteam
GUI: 1 tim, logic: 1 tim

Workflow 3 - HR

Request staff

As SM/PM I should be able to request new staff from HR
GUI: 30 min, logic: 40 min

Manage staff requests
As staff in HR I should be able to see staff requests and write how it was solved
GUI: 40 min, logic: 40 min

Manage staff requests Split from “Manage staff requests”
As staff in HR I should be able to describe how a staff request was solved
Time estimate already in base story

Workflow 4 - FM

Create financial request
As SM/PM I should be able to create financial request and send it to FM
GUI: 30 min, logic: 40 min

View my financial requests
As SM/PM I should be able to see my submitted financial requests
GUI: 40 min, logic: 50 min

Manage financial requests
As FM I should be able to view financial requests and choose whether to accept or deny them. Alternatively add a reason for the decision
GUI: 30 min, logic: 40 min

Release 1

User Story Name	Value	Risk	Iteration
Login	H	M	1
Create client	H	L	1
Create event request	H	L	1
Create tasks	M	H	1
View tasks	H	L	2
Manage tasks	M	L	2
Create staff request	M	M	2
View staff requests	M	L	3
Create financial request	H	M	2
View financial requests	H	H	3

Release 2

User story name	Value	Risk	Iteration
Review event (Financial)	H	M	1
Review event (Administration)	H	M	1
Forward event requests	H	L	1
Request staff	M	M	2
Manage staff requests	M	M	3
Manage financial requests	M	M	3
View my financial requests	L	M	3
Comment on task	L	L	3

Emil Karlsson & William Asp

Metaphor

Workflow 1

Production line	**SEP Creating an event**
Customer	Input resources
Analyzing machines	Reviewers
Resources splitter	Production manager
Constructor machine	Subteam

Resources are analyzed to determine its value	Reviewer go through the event request to see if it's feasible/doable

Workflow 3

Car Workshop	**SEP Tasks in subteam**
Customer	Production Manager
Mechanic	Some subteam

Customer informs about what needs to be fixed with the car	Customer informs what kind of the desired event
Mechanic find more problem with the car that needs to be fixed, which costs more	A subteam notifies in the task’s comment about changes in budget

Emil Karlsson & William Asp

Acceptance Test

Test Case Name	Login
Expected Action	1. Open the CLI application 2. Enter username “test” 3. Click Enter 4. Enter password ‘password’ 5. Click Enter
Expected Result	User “test” is logged in
Test Result	Successful

Test Case Name	Redirect to correct UI
Expected Action	1. Login with Production manager Jack 2. See correct Production Manager interface
Expected Result	User ‘jack’ sees only options that he is allowed to see, create tasks, manage tasks
Test Result	Successful

Daily stand ups

Meeting date	October 5th
Participants	Emil, William
Meeting Notes	<u>Today expected actions</u> a. Create the type script node app b. Write skeleton code for login c. Add department checks when logged in
Comments	 Written by Emil

Meeting date	October 6th
Participants	Emil, William
Meeting Notes	a. Add database for users b. Make login check user credentials from database when logging in
Comments	 Written by Emil

Emil Karlsson & William Asp

Summary

Implementation

Test-driven programming proved to be useful in different ways. Partly because it showed a clear way what needs to be implemented, but also since no test needed to be come up with afterwards, as this could lead to missing some faulty branches in the code. This worked well from a pair programming standpoint as tests are easy to discuss and refactor early on.

The first and major refactoring was the introduction of a database. This was a technical aspect of the application, and thus not directly visible in the user stories. After login was firstly implemented as a simple “if user is this user and password is this password”, a database interface abstraction was added to facilitate more changes in the future (even if the application runs in memory).

Finally, the time specified in the user stories was mostly exaggerated. In reality, most of the user stories were implemented much quicker.

Comparison

Object oriented analysis focused a lot on preparations for a system to eventually be built. It was beneficial to show how a system will look like and how the expectations will be modeled. On the other hand, it consumes a lot of time to create the documents and go through the application more or less only in theory for the entire time. It gave no way for the developer to show any kind of meaningful prototype during the design process apart from diagrams with some figures and text.

Using Extreme programming a much more pragmatic approach was used. Take a list of what the user wants and implement them. While this approach proved to be far easier to wrap your head around since less documentation language is needed, it still required skill and thought when implementing application in code. Since no real plan on how different user stories will be implemented alongside each other, it was vital to do a very good job the first time. In one way, the plan and roadmap was always in the head of the developer. Further, to remedy the client to developer communication, metaphors were created. While this was aimed at simplifying client communication, it was beneficial to view the system as something else in order to better grasp the different subsystems responsibilities.