P2P FILE SHARING

‘The Octopus’ Group  
  
Team Members:

AHMAD, FARHAN

ALIPOURSIMAKANI, KAMRAN

ANDERSSON EKSTRÖM, MAX

BERNTSSON, FREDRIK

CHADALAPAKA, GAYATRI

IQBAL, NAYYAR

KUKKAPALLI, NAGA VYSHNAVI

NYHLÉN, JESPER

ROUTHU, VENKATA SAI KALYAN

SHAD MANFEAT, SEYEDEH MERSEDEH

ZAREI, KAMBIZ

Type of Document: Project specification

Version 1.2

Publication Date: May 15th ,2016

**1. PREFACE:**

***Section 2*: Glossary and abbreviations**

***Section 3*: Background**

***Section 4*: Proposed Solution**

***Section 5*: Limitations**

***Section 6*: Time Plan**

***Section 7*: Project Organization**

***Section 8*: Configuration Management**

***Section 9*: Progress Tracking**

***Section 10*: Quality Control**

***Section 11*: Risk Management**

***Section 12*: System Release Plan**

***Section 13*: References**

**Release v1.2 on 2016-05-15**

-clarified sprint duration in time plan

-specified date in system release plan

-updated reference list with product request

- update section 9, filled in more text to all areas and added meeting notes

- update section 12, added sprint dates and sprint goals and what the status are on them.

**Release v1.1 on 2016-05-01**

-updated format of document

-updated timeplan

-updated project organization to describe our procedures

-clarified the numbers in risk management

-improved timelines in system release plan

**Release v1.0 on 2016-04-17**

- Initial release

**2 Glossary and abbrevations**

Java - A general purpose object oriented programming language

Apache Maven - automated build tool for java code. Build the source code to a working “.Jar” file.

JUnit - Small explicit test to see that the code's functionality does what you think it does. For example 3+3 should generate 6 if it does it pass, if it generates another number it fails.

Git - A tool that keeps track of changes in the source code and lets people collaborate.

Gitlab - A program that has the Git functionality for the users and then uses a website with additional features. In order to let people collaborate with git it is needed to be run online on a server.

Dark Peer - The peers who first connected to the bootstrap server and get a list of peers from the bootstrap server. After its validity time, it doesn’t talk to the bootstrap server anymore.

Dark Content - The file contents which exists on the dark peers

Swarm - The main file which each peer gets from the server at its first connection which includes the shared files, the list of peers who shares the file and the swarm metadata.

Swarm Metadata - Includes file names, file message digest

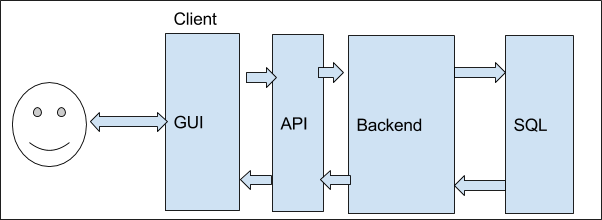
File Metadata - The set of headers together with the filename is the file metadata.

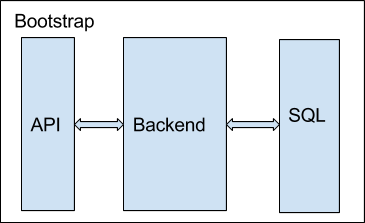
Bootstrap Server - the main server which new incoming peers connected to and get their swarm metadata from.

**3: Background**

“The customer P2P4FUN requests to design a simple yet efficient P2P file sharing application which should satisfy its needs as follows:  
Upon start, the peers connect to the bootstrap server and they receive a list of published swarms and a list whit banned IP addresses are which they should refuse to communicate with.  
Each peer is responsible to refresh the validity time for its IP address; IP addresses that exceed the expiration date are removed from the bootstrap server.  
There will be at least one backup bootstrap server. If no bootstrap server is available, the peers use the current data they have and will receive a message telling them about no bootstrap server where available.  
 Each peer must maintain a maximum of 3 connections to other peers.  
the P2P application must have a friendly graphical user interface (GUI) providing all necessities to create, list, and delete swarms, and to search, upload and download files.  
In addition, there should be a way to select if a peer goes dark or becomes visible again.  
Progress of uploads and downloads must be shown together with estimated time to complete. The GUI must also show the IP addresses associated with a swarm and the IP addresses of the connected peers.  
The application must also keep track of upload and download bandwidth and show this as live time-series plots.  
P2P4fun insists that all communication must be encrypted. Plus, It should be possible to restart the system with encryption disabled for debugging purposes.  
Finally, P2P4fun requests that all interaction user-to-user and user-to-server be based on a RESTful API with JSON data encoding.  
 your product supports at least 3 peers.” [1]

**4: Proposed solution**





In order to make the software flexibility for changes it will have loose couplings between frontend and backend. The frontend is the GUI (Graphical user interface) and the backend is where all the calculations and program logic is. The communication between the backend and the front end will be through a simple API which will make it as lego bricks, easy to change and replace as long as the format is followed.

Communication between the client and the bootstrap server will be through REST - calls over HTTPS where the data will be encoded in JSON. Also the communication between client and client will be with REST and encoded in JSON (peer to peer traffic). The communication will be from the API on the client to the API on the bootstrap, the “API” is the face outward for the program.

Java programming languages will be used for the client and bootstrap server.

We connected our bootstrap to our API, to define swarm metadata or file metadata like our peer lists (as whitelist or blacklist) and other functionalities which can be managed easily from the API.

A database will be used to store the swarms created by peers, the file metadata and the servers metadata. **5: Limitations**

The bootstrap server won’t allow peers to download the file from the server just the metadata that is needed for downloading from peers.

The client will be platform dependent and run on windows and if possible also Ubuntu.

The client will support 3 peers using the service and if possible even more.

There will only be one Bootstrap server as backup.  
 The service will not be available all the time.

The client and Bootstrap server will be in English only.  **6: Time plan**

We will be having 6 sprints which are exactly based on the deadlines of our assignments.

Sprint 1: 11/4-17/4

Sprint 2: 18/4-24/4

Sprint 3: 25/4-1/5

Sprint 4: 2/5-8/5

Sprint 5: 9/5-15/5

Sprint 6: 16/5-22/5

We are going to break down our project to smaller tasks and by using ProjectLibre (or at the same time using a Kanban board) we will choose tasks, analyse, develop and test them accordingly!

**7: Project organization**

We will be using agile development model:

-workload will be divided between 6 sprints.

-tasks will be written to complete each sprints work.

-weekly meetings to report between teams, discuss progress, decide on next sprints work, and solve problems we currently have in project.

-During the project, we will simultaneously work with coding, testing, documentation and management. We will split the group up into several sub-teams.

-All project documents will be updated during each sprint, these include Project specification, Software requirement specification(start sprint 2), Design specification(start sprint3) and Acceptance test plan(starting from sprint 4)

-

|  |  |
| --- | --- |
| WORK | TEAM MEMBER |
| 1. Programming:   A. Frontend:     B. Database:  C. Backend: | ALIPOURSIMAKANI, KAMRAN NYHLÉN, JESPER ROUTHU, VENKATA SAI KALYAN KUKKAPALLI, NAGA VYSHNAVI IQBAL, NAYYAR  AHMAD, FARHAN CHADALAPAKA, GAYATRI IQBAL, NAYYAR  BERNTSSON, FREDRIK ZAREI, KAMBIZ  ANDERSSON EKSTRÖM, MAX SHAD MANFEAT, SEYEDEH MERSEDEH |
| 2. Management: | BERNTSSON, FREDRIK ALIPOURSIMAKANI, KAMRAN ZAREI, KAMBIZ  NYHLÉN, JESPER |
| 3. Testing: | BERNTSSON, FREDRIK ANDERSSON EKSTRÖM, MAX  ALIPOURSIMAKANI, KAMRAN NYHLÉN, JESPER ZAREI, KAMBIZ |
| 4. Documentation: | ROUTHU, VENKATA SAI KALYAN NYHLÉN, JESPER GHASEMI REZAEE, AMIN KUKKAPALLI, NAGA VYSHNAVI |

**8: Configuration management**

**8.1 Version management**

For version management of our coding, Git will be used on a Gitlab server. The reason that we choose Git is in order for everyone in the team to be able to access and share the latest code easy. Also Git will keep track of the code history and who has written which part of the code.

For our project documents, we'll use google drive. This makes it easy to instantly see each others changes and work simultaneously on the same document.This also makes sure there will be no conflicts and everyone can see the progress of the documents.

**8.2 System building**

Apache Maven will be used for system building, the reasons for this is that the code will be in Java and Maven will automate the building process and with a POM file it will fix all the dependencies. A big pro with Maven is that in order to build the project it has to pass the JUnit test (See section code quality), this will make sure that the code uploaded to the git is not broken.

**8.3 Release management**

Our release management begins in the development cycle.If our customer request us for some new features or changes for instance. If the request is approved, the new release is planned and designed. The new design enters the testing or quality assurance phase, in which the release is built, reviewed, tested and tweaked until it is ultimately accepted as a release candidate. The release then enters the deployment phase. There will be changelog for every release which will contain bugs fixed, new features and things that were improved.

**9: Progress tracking**

After creating the baseline schedule, it's important to track the progress of work on the project. Since different sub-groups will be created and tasks will be assigned to each group. For tracking purpose, the team use ProjectLibre, Gitlab and meetings. In Projectlibre is more on a high level that is use cases and in Gitlab is on more specific task within the use cases and on the meetings we speak about what did go wrong and if people are having more practical problem.   
  
**Projectlibre** has a high overview of the project when the use cases need to be done in order to not delay the project.

**Gitlab** has more specific task that is defined in different milestones, those task is assigned to different team members. The task is clearly linked to the milestone goal that is written on gitlab. Goals for each milestone is linked to the project libre overview of the project that is, those things needs to be done on this date in order for the project to be on time track. For each milestone in Gitlab it is possible to see how many percent is done and when the milestone is ending. Different task is given different labels, labels determine where the issue is fitted in the project for example in Gui, backend or database. There is also labels that determine if the task is of high priority or low priority.

**Weekly meetings** we talk about what have gone good this week and what is needed to be changed. If people has questions or problem it is a excellent opportunity to ask them here. The meetings is around two hours. Each member also has to speak about what they have done since the last meeting and what problems they have occurred. It is also in this meeting the direction of the project is determined and what we need to focus on in next week.   
**Dates for the meetings has been:**

* **25 - April**Members are reading up on java. Client and bootstrap has been started developing. Parsing between POJO -> Json -> POJO.

* **27 - April**Make a scheme for future meetings. Improve how to split task, every one should be able to make tasks. Bring all the members of the team to the same level of the project. Clear instructions of what each member should do. Decided to kick out Taiga, too much work. Started using Gitlab instead.
* **2 - May**Most members have sat up their development environment. Research of different areas has started. Green light on database team, backend and GUI. Red light on documentation. Not all members that should be working on it those it. Junit testing can be started on client. Moved git repo to final repo. Java threads have been done, rest calls are working. Gui compiles.
* **9 - May**Got information that some members don’t have trained in java has dues have not the required knowledge to contribute to project in development right now. Database is talking to backend. Mocked “swarm” data in database. Maven built updated. Started refactor code and comment code. Talked about what should be done when members are not showing up for meetings and not contributing to project.
* **15 - May**

**10: Quality control**

In order to have a high quality of the code there will be regularly code reviews especially on complex functions. Commenting will be in english and common code standards will be followed. There will be task for each user requirement due to milestones that will be tested. Junit testing will be made where it is possible to make sure that functionality is working as expected.

**11: Risk management**

Risks will be measured from 1-3, where 1 is low, 2 medium and 3 high.

|  |  |  |
| --- | --- | --- |
| **Risk** | **Likelihood (1-3)** | **Significance (1-3)** |
| **1. Management** | **3** | **3** |
| **2. Group room** | **3** | **2** |
| **3. Communication** | **2** | **2** |
| **4. Lack of knowledge** | **2** | **2** |
| **5. Sickness** | **2** | **1** |
| **6. Moral** | **1** | **3** |

1. **Management is the key to success, we are twelve persons so if not everyone has something to do all the time we will lose lots of time. In order to avoid losing time we will use kanban board with tasks that each member can grab.**
2. **Group room will be a big problem because it will makes the communication harder. But also that it’s hard to have a real schedule for work and working home is not as efficient. We will minimize the risk with using task tracking tool (Taiga) and time tracking tool and use a centralized repository for the code, Gitlab.**
3. **Communication, The team has many nationalities and not the same mother tongue. The communication will be done in english and it is easy for misunderstandings. To minimize the risk we will have many face to face meetings.**
4. **Lack of knowledge, the area we are working in is new for many in the team and not everyone is comfortable with the practical part of programming. The security part is new for almost everyone and also working in this large groups.**
5. **Sickness with team will make us lose valuable time and knowledge. To minimize the risk we will have meetings to keep track of what each person does and two people that should know about the areas.**
6. **Moral will be a problem if the team get stuck. In order to minimize the moral troubles we will bring fika if the moral goes down and talk about it.**

**12: System release plan**

The team has determined to have one week sprints that goes hand in hand with the milestone goals. The sprints are managed in gitlab.

**Sprint #3 *24/4 - 1/5* *Milestone 3***

* Client to client send hardcoded file **(Done)**
* GUI that compiles for the client **(Done)**
* SQL communication works **(Done)**
* README in tip-top condition **(Done)**

**Sprint #4 *2/5 - 8/5* *Milestone 4***

* GUI presents real data
* The database is working
  + Safe from SQL injections
  + We use it in the client and bootstrap **(Done)**
* HTTPS is used to some extent **(Done)**
* Threads are used **(Done)**
* Create a swarm from client to bootstrap

**Sprint #5 9/5 -15/5 Milestone 5**

* Client GUI fully working
* HTTPS is used for all REST calls
* All bootstrap REST calls are working **(Done)**
* All client REST calls are working **(Done)**
* Unit tests are done
* Download file from multiple clients

**Sprint #6 16/5 -22/5 Milestone 6**

* Integration tests
* Code refactoring
* Crisp GUI
* Bootstrap GUI

8/5-2016: Alpha release

15/5-2016: Beta release

22/5-2016 : Final release

**12.1: Testing plan**

Code testing: Junit will be used to test our code, which will give a clear overview of the test results every time you build the system.

System testing: System testing phase will begin during sprint 5 and end sprint 6.

Integration tests: Integration testing will start in beginning of sprint 5.

Acceptance testing: We will be doing acceptance testing starting from sprint 5 to ensure that our product meets the requirements.

**12.2: Packaging plan**

A compressed tar.gz archive is available to the user which includes the code, library files, related documentation and tools.

**12.3 Installation Documentation**

Installation documents will be delivered in PDF format and contain information on how to install and configure our software. This will be done for the alpha-release of the software, and updated if changes occur to the installation process in future releases.

**12.4: User Documentation**

The PDF format is used to release User documentation. It includes the scope of the tool, tool functionality, linking different modules in the tool, process of supplying inputs to the tool, generation of output and various scripts. The user documentation will be worked on during sprint 6.

**12.5: Developer documenation**

This Developer document includes about the scope of the project in future. This helps to develop the sources code and necessary information about the GUI’s used in this project. Javadoc will be used to provide a html documentation of our java code. An initial version of this will be provided in sprint 4, this version will be improved during sprint 5 and 6.

**13: REFERENCES**

[1] P2P4fun product request, P3: P2P file sharing, 2016. [Online]. Available at; https://bth.itslearning.com/LearningToolElement/ViewLearningToolElement.aspx?LearningToolElementId=639246, (Accessed: 2016, May 15)