**Functional specifications**

**Cubbyhole 2014**

1. ***The Cubbyhole solution***
   1. *Presentation*

**Cubbyhole** is a free service that lets you bring your photos, docs, and videos anywhere and share them easily. Never email yourself a file again!

This file hosting service operated by our team and Cumorah Interactive will offer cloud storage, file synchronisation, and clients software through cross-platform.

* 1. *The team*

The team is composed of five engineers :

Moriceau Mathieu - SysAdmin / Developer

Gourgouras Sebastien - SysAdmin / Developer

Vialleton Anthony - Architect / Developer

François Guillaume - Server Side Developer

Moreno-Semedo Igor - Advanced Developer

* 1. *Tools and access*

In order to work efficiently our team is hosting its solution mainly using Github through private repositories for code sharing and a dedicated Redmine web client interface to manage our tasks and monitor them.

See the link below : [https://github.com/](https://github.com/francog06/cubbyhole)watcheur[/cubbyhole](https://github.com/francog06/cubbyhole)

* 1. *The goal*

The main goal is to offer a high value services to final customers with a complete, fair-use and high-available solution from everywhere and for everyone !

1. ***Web client specifications***
   1. *PHP overview*

**PHP, HTML5/CSS3** and **Javascript** are very easy to use, deploy and are very powerful, *we use some libraries like JQuery, Bootstrap and JqPlot to create a fancy web interface.*

Further PHP is well documented and has a lot of well-done framework.

**Sublime Text 3** will be mainly used as IDE because of his high quality plugins native integration.

* 1. *Corporate website*

The first part of the web client consists of a commercial website that allows users to **register, pay and manage** their **subscriptions/plans** through an user friendly interface.

* 1. *Users accounting and plans*

Users can create accounts from the website and select a plan.

Only a **mail address and a password** will be required to register.

These users will be able to manage their associated plans across a very simple **Boostrap interface**.

Plans represent access levels and matching rights and will be feed by a **PHP Paypal module (\*\*)**.

Three plans will be made available to customers **(\*)**:

• **Free plan**:

* 0 euro
* No expiration deadline
* 1 Gb Usable storage space
* 100 kbps Max bandwidth
* 100 MB shared link daily transfer

• **Pro plan**:

* 2 euros
* Monthly expiration
* 50 Gb Usable storage space
* 512 kbps Max bandwidth
* 512 MB shared link daily transfer

• **Illimited plan**:

* 9 euros
* Monthly expiration
* 100 Gb Usable storage space
* 1 Mbps Max bandwidth
* 1024 MB shared link daily transfer

**Plans disk quotas and bandwidth** have to be always considered and it will very **simple for an user to fully change and mangage his plan (\*\*\*\*).**

* 1. *Web client features*

An **user** will be able to :

* **Manage** (**create,move,copy,delete**) **files and folders (\*\*\*\*)**
* **Download files (\*)**
* **Upload files (\*)**
* **Share** **files** with **Cubbyhole users (\*\*\*\*)**
* **Set per user** **read only or read write sharing permissions (\*)**
* **Modify share permissions and revoke sharing (per user) (\*)**
* **Generate share links for non Cubbyhole users (\*\*)**
  1. *Cubbyhole business dashboard*

In order to look more closely the Cubbyhole evolution our administrators will have access to a **well designed backend**: the business dashboard.

This dashboard will be composed of **relevancy indicators** **(\*\*\*)**:

**- User statistics bar diagram** : number of free users, paid user, registration date, plans.

**- Usage statistics graphics** : plan specs usage (used space/paid space, used bandwidth/paid bandwidth,  etc.)

- Where are users located using **IP Geolocation** ?  To answer questions like and provide % indicators such as:

- How long it takes for a free user to become a paying user

- How much free users decide to buy a plan

- Are people using 100% of their plans

- Will a more expensive unlimited plan be more profitable than a less expensive but more  limited plan?

An **administrator** will also be able to :

**- Generate reports of one metric against two others (\*\*\*\*)**

- **Compare report against each other** **(\*\*)**

**Clarity** and **informations** will be the main purpose of this feature.

1. ***Web service API specifications***
   1. *Centralized data*

**The Cubbyhole** web service API will be developed in **PHP** and will be hosted in a dedicated server.

This server will also host and manage the centralized **MySQL 5.6** database wich will be able to feed all the clients of Cubbyhole solution.

* 1. *Frameworks*
     1. *CodeIgniter*

**CodeIgniter** is an Application Development Framework - a toolkit – to build web services using **PHP**. Its goal is to develop project much faster by providing a rich set of libraries for commonly needed Restful tasks, as well as a simple interface and logical structure to access these libraries.

ii*. Doctrine*

**Doctrine** is an **ORM** (Object Relational Mapper) library in PHP, stable and supported since 2006, it is very flexible and powerful, and it can be integrated with many different frameworks.

* 1. *Web service API features*

*Users are able to get informations through our API, like their account, the folders and files they share, their plan and data history.*

d. *Conclusion*

*Our choice is to build an API the easiest way, light, flexible, powerful with full support, that's why we choose to couple CodeIgniter and Doctrine.*

*For our team, the M.V.C (Model View Control) model from CodeIgniter and the ease of use of the Doctrine library for the development was the best choice.*

1. ***Mobile client specifications***
   1. *Android devices overviews*

**Android** powers more than a **billion phones** and tablets around the world. *It’s customizable, yet easy to use because of the* ***Java*** *language used for development and* ***Eclipse*** *powerful integration.*

* 1. *IOS devices overviews*

***IOS*** *is the operating system that powers iPhone, iPad, iPod Touch, and Apple TV, applications are developed in Objective-C and the Xcode IDE.*

*This platform represents about more* ***then 65% of the mobile OS market*** *today.*

* 1. *Features*

A **user** will be able to :

* **Manage** (**create,move,copy,delete**) **files and folders (\*\*\*\*)**
* **Download files (\*)**
* **Upload files (\*)**
* **Share** **files** with **Cubbyhole users (\*\*\*\*)**
* **Set per user** **read only or read write sharing permissions (\*)**
* **Modify share permissions and revoke sharing (per user) (\*)**
* **Generate share links for non Cubbyhole users (\*\*)**

*d. Libraries*

*We use some libraries for Android devices:*

*-* ***aFileChooser and DirectoryChooser****: It provides simple and great files and folders pickers depending on the OS version.*

*-* ***AppCompat v7 (Google)****: It provides Android 3.0+ design features for Android phones using the 2.1 minimum version provided by Google, to build fancy interfaces.*

*-* ***Volley (Google)****: It's a network library for Android phones provided by Google, to make lightweight and efficient HTTP requests, writing less code and focus more on the business code development.*

1. ***Win32 desktop client specifications***
   1. *C# .Net overview*

C# is an elegant and type-safe object-oriented language that enables to build a variety of secure and robust applications that run on the .NET Framework. This can be used to create Windows client applications (Win32), XML Web services, distributed components, client-server applications, database applications, and much, much more.

This desktop client will be developed using **C# .Net and SQL Server and Visual Studio 2013**.

* 1. *Synchronisation client features*

**The synchronization** client only **blindly synchronizes (back and forth) the user Cubbyhole on the local machine.** However, the user should be able to **select** which folders **he (doesn’t) wants to synchronize**.

So the two main features are :

- **Synchronize** the Cubbyhole **locally** : **download & upload** **(\*\*\*\*)**

- **Users** can **select which remote folders should be sync locally** **(\*\*)**

***6. Cubbyhole network architecture***

* 1. *Architecture listed features*

Architecture main features :

- Network design: **link redundancy, bandwidth management** **(\*)**

- The **application servers** are **highly available and load balanced** **(\*\*\*\*)**

- The **web client servers** are **highly available and load balanced** **(\*\*\*\*)**

- The **storage** is **highly available and load balanced** **(\*\*\*\*)**

- **Monitoring** : Administrators are **notified** if something goes down **(\*\*)**

- **Health monitor** **(\*)**

* 1. *High availability*

**The application servers**, **the web client servers and the storage are highly available and load-balanced**.

* 1. *Application server*

**Application servers** run the **core application that provides API services to other clients**. There must be more than one of these machines, and it should be **simple to add one to the pool.** These machines must **share the load from various clients and failover if one of them goes down**. They in turn use the storage backend to fulfill the client requests.

* 1. *Web client*

**The web client** is the main client: It provides both a commercial website and a file manager. It should run on **separate servers** and can be written in any language, not necessarily the same as the application itself. The web client **cannot run on a single machine**: the load should be distributed across several machines that also provide failover.

* 1. *Storage backend*

**The storage backend** is responsible for storing the actual data. **It also can’t be a single machine’s task:** If that machine were to go down, the whole system would also go down. Feel free to use any solution you find fit: **iSCSI cluster** with replication with the nodes, etc. The only requirement is that actions from the application servers should not fail.

This server will be used for **storage backend and Restful web service API hosting**. The architecture splitting wille increase performances and make this API get the full power resources she needs.

* 1. *Internal network paths, monitoring and administration*

**The internal network paths** between **the servers, the backend and all the machines** that belong to the supporting architecture is very important. The purpose is to design a **robust and scalable “internal” network** (router, switches, failover links, etc.) for the solution.

This structure should be **monitored and administrators** should get alerts on their mail and mobiles whether something in the architecture (network, servers, storage, ...) should go down. The monitoring solution should also provide a health monitor that shows instant status of the whole service.

1. ***Labours management***
   1. *Redmine*

**Redmine** is a flexible project management web application. Written using the Ruby on Rails framework, it is cross-platform and cross-database. Redmine is open source and released under the terms of the GPL wich is the best solution to manage our labours with value for money.