Faculty of Natural & Mathematical Sciences

Group Project - 7CCSMGPR
Deadline Fighters

Preliminary report - February 8, 2019

Sivaranjani Subramanian
Ka Hang Jacky Au Yeung
Lili Chen
Letao Wang
Qilin Zhou
Georgii Fidarov

1 Aim

Our team, *Deadline Fighters*, aim to develop a multi-host file synchronizer which can:

- allow upload and download of files from a central server (hub) through client applications with necessary authorization.
- automatically synchronize changes made by client applications unto the corresponding copy of the file in the server.
- handle all possible conflict/non-conflict scenarios of file synchronization (ie. combinations of Create, Edit, Rename, Delete) involving multiple client applications.
- enable file sychronization through both desktop (all platform) and mobile (Android) clients.

2 Strategy

High level workflow

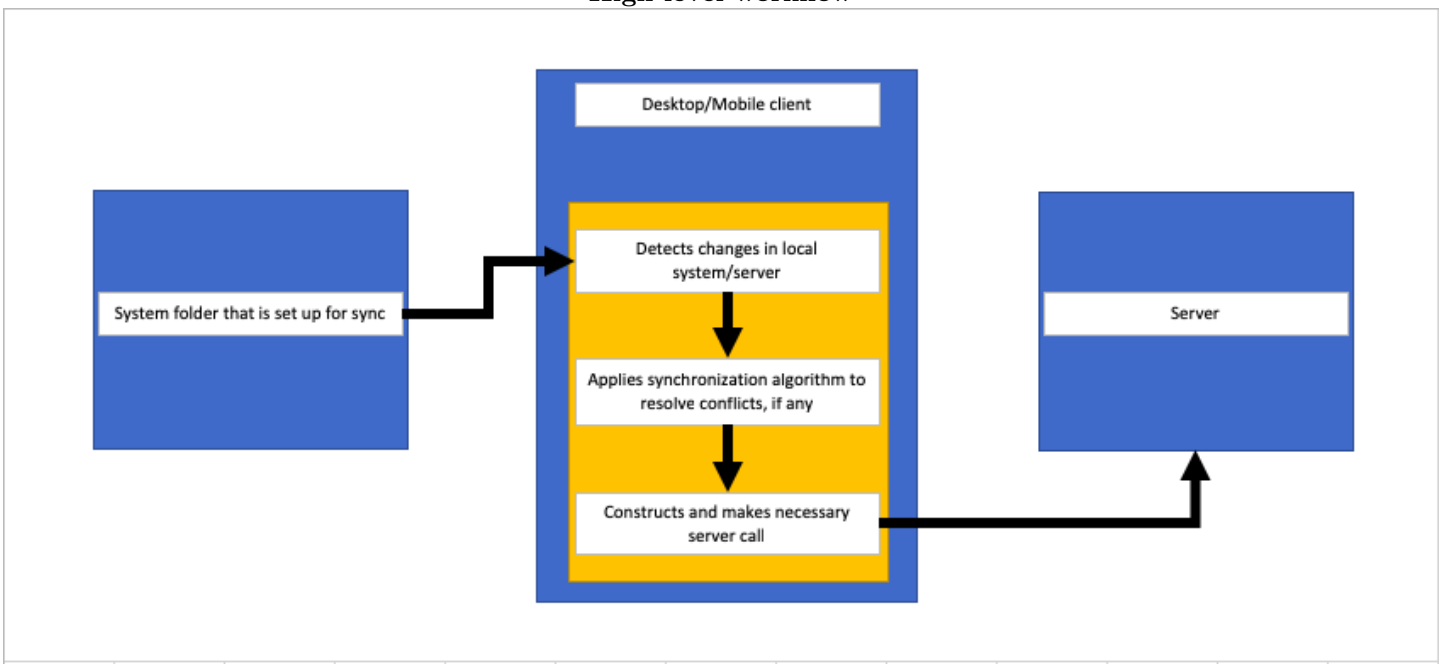

Figure 1: Flow of control between client and server

In our Agile workflow, we will start by building a basic prototype of the desktop client. This prototype will perform functionalities of Create, Edit, Rename and Delete for non-conflicting files for single user. It will also have a *Upload* and *Download* button to manually invoke synchronization. Then we will extend this prototype to handle conflict scenarios. The potential conflict scenarios and their resolution are stated in table 1.

Once the conflict resolution algorithm implementation is stable, we will develop the mobile client prototype followed by testing conflict resolution using mobile-desktop client pair. After satisfactory testing, we will move to the final phase of introducing automatic file sync using *file system notification* and *server polling*. Note: We are still contemplating on whether automatic sync should be a desktop-only feature due to constraints on mobile such as battery-life, available internet allowance etc.

- **Milestone 1**: Prototype of desktop client for single user with manual sync.
- **Milestone 2**: Prototype of conflict resolution on desktop client for multiple users with manual sync.
- **Milestone 3**: Prototype of mobile client for multiple users with manual sync.
- **Milestone 4**: Enable automatic sync on desktop client.

3 Tools/Technology

We will use the following tools and technology for the corresponding purposes:

- **Amazon S3**: We use S3 as our cloud/server endpoint. S3 provides APIs for Javascript and Java which we will exploit for our desktop and mobile client respectively. S3 handles encryption and notifies us of any potential vulnerabilities.
- **Electron/Javascript**: For building the desktop application. It uses *Chromium* and *Node.js* on the backend and helps convert web application code (HTML/CSS) into desktop app.
- **Android Studio**: For building mobile client.
- **MySQL**: For Database. Note: We have not investigated other possibilities in depth at the moment and thus this choice is open for changes.
- **Charles**: For web debugging. This will help us debug server calls as well as help simulate conflict scenarios.

- **Mocha & JUnit**: For writing unit tests on Javascript and Java respectively.
- **GitHub + Git Bash/terminal**: Version control and code collaboration. We have created an organisation called *deadlinefighters* of which everyone is a part of. Everyone has write access whereas *Sivaranjani* and *Letao Wang* have administrative privileges. We will also use Git for bug tracking and pre-merge (build) checks.
- **Atom**: We use Atom as our code/text editor as it has Git and LaTeX compatibility, giving us a single view of our important tools.
- **LaTeX**: We use LaTeX for documentation and our LaTeX file is also collaborated on GitHub. Every team member has LaTeX compiler installed in their system along with a plugin for LaTeX on Atom which helps compile while on the text editor.
- **MockingBot**: MockingBot is used for the designing user interface of our desktop and mobile client. This will help streamline developer efforts.

4 Team protocol

Our software engineering methodology will be agile, which means that our development is iterative and spread across *sprints*. The project divided into several milestones, each will be the aim for one or more sprints. Every sprint lasts for one week and involves requirement collection, planning, design, development, testing, deployment and review.

There are two types of meeting that the team will be held:

- The first type is held every week and is attended by all members. During this meeting, team members will review the work done last week, discuss the problems encountered as well as potential problems and plan for the week ahead. This meeting usually lasts for 1 hour.
- The other type is held every 1-2 days. This is not for all members but only for members who have the same/related tasks assigned in the current sprint. This meeting usually does not involve major decision making but reviews members' progress since last meeting. This is our *scrum meeting* and lasts no longer than 5 minutes.

Other rules framed on consensus include:

- All team members should push at least one commit every three days.
- Once a member raises a pull request on the git, at least 2 members should view the changes and approve it.
- If some members request changes, the member who raised the pull request make necessary fix or give a satisfactory explanation to not incorporate the changes.
- In case of clash of technical opinion among team members, voting or some form of detection of majority will take place and their stance will be implemented. In case of personal conflicts, team members are expected to resolve among themselves and can ask the co-ordinator to step in if necessary.
- There are two feature branches: *desktop-client* and *mobile-client* where desktop and mobile client development will be tracked respectively. On achieving some milestones, we will merge these feature branches into master branch.
- Every team member who introduces new code will also author unit tests and relevant documentation as necessary.
- **Definition of feature complete**: A feature is marked complete when the necessary coding work is done, thoroughly tested and has 0 major/critical bugs and necessary documentation is complete.

Modes of communication: Face-to-face meetings in the open area in Bush House (N) 6th floor, WhatsApp

5 What we have done within recent weeks

Totally we have 10 meetings within 3 weeks. We had our first meeting to get to know each other and finalise on group name and team coordinator. All members registered on the GitHub and completed initial tasks as set by Dr.Tratt. In our second meeting, we analysed what our tasks are and distributed research tasks on server, unit tests, mobile/desktop dev, file sync and conflict resolution algorithm amongst team members.

The following meeting, some decisions on the technology to be used were made. In this meeting, we moved from Amazon EC2 to S3.

In the following meetings, we started working on our desktop application as well as the initial report. Sections of the report was allocated to each member who worked individually and pushed to git. The team co-ordinator then made some overall refactoring for the final report. Also, Lili Chen designed a sample user interface of the desktop application using MockingBot.

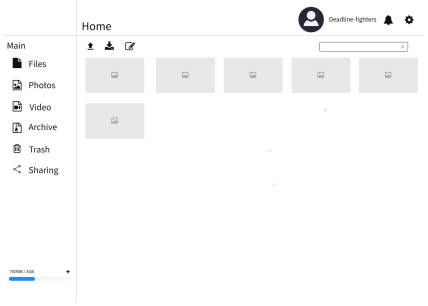

Figure 2: Mock UI for desktop client

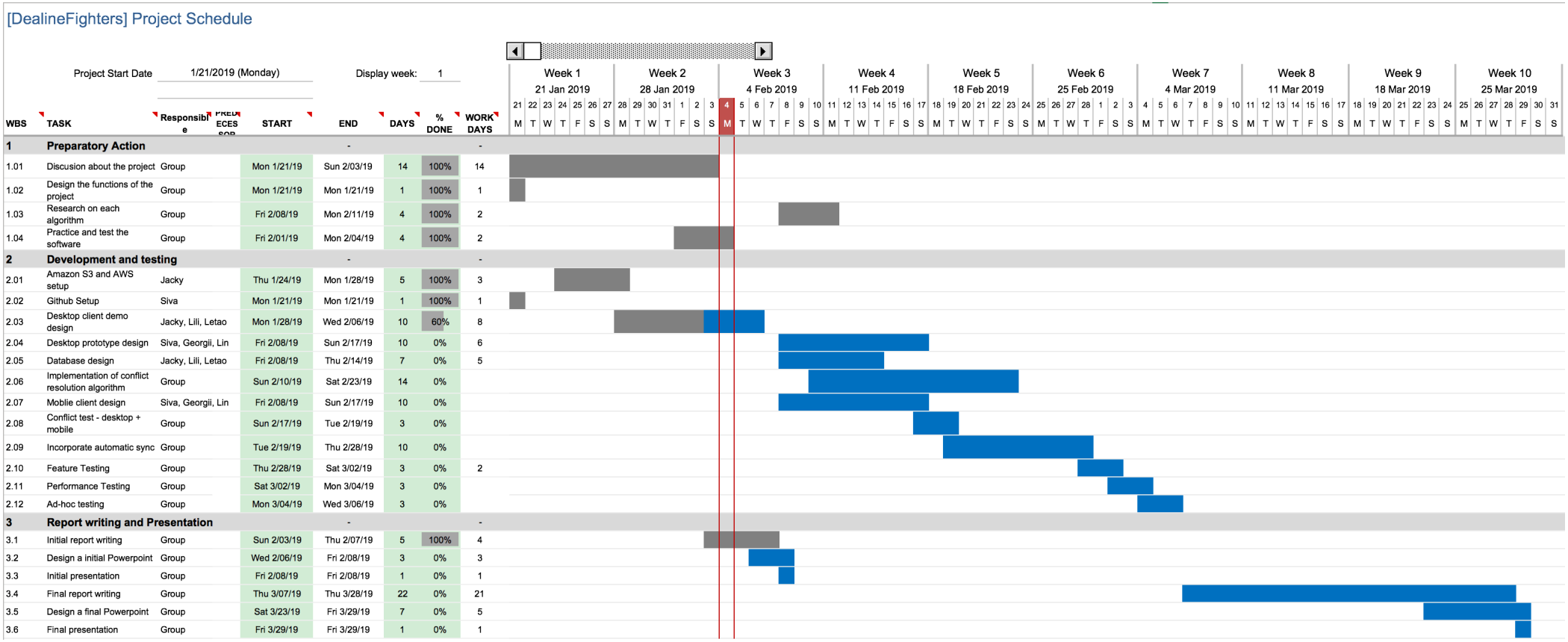

Figure 3: This is Gantt chart which illustrates the plan of our group within 10 weeks.

Table 1: Conflict scenarios and resolutions

No.	Operation 1	Operation 2	Resolution
1	Create	Create	Upload both files, one with the original name and one with a prefix (eg. (1) or 'copy')
2	Create	Edit	Upload edit changes to the original file and upload new file with a prefix (eg. (1) or 'copy')
3	Create	Rename	Rename is applied first followed by upload of new file
4	Create	Delete	Delete file first and then upload the newly created file
5	Edit	Edit	Possible strategies: • Send server error for both requests • Upload two copies of the file, each with different edits
6	Edit	Rename	Rename original file and upload document with edited changes as a new file
7	Edit	Delete	Delete original file and upload document with edited changes as a new file
8	Rename	Rename	Rename original file on server to name1 and upload another copy of the file with name2
9	Rename	Delete	Delete original file and upload another copy of the file with the new name
10	Delete	Delete	No conflict

Operation 1 and 2 are performed on/for files with the same name by same/different users from different devices.

References

[1] GitHub, [online]Available at: `https://github.com/`(Accessed: 3 February 2019).

[2] ElectronJS, [online]Available at: `https://electronjs.org/`(Accessed: 3 February 2019).

[3] Amazon S3, [online]Available at: `https://aws.amazon.com/free/storage/?nc1=h_ls`(Accessed: 3 February 2019).

[4] Charles, [online]Available at: `https://www.charlesproxy.com/`(Accessed: 3 February 2019).

[5] Mocha, [online]Available at: `https://mochajs.org/`(Accessed: 3 February 2019).

[6] Junit, [online]Available at: `https://junit.org/junit5/`(Accessed: 3 February 2019).