# Milestone 3: Final Project Proposal (group work)

- Mike Hoffert - mlh374
- Syed Ahsan Rizvi - sar457
- Hattan Alsharif - haa775
- Da Tao - dat293
- Michael Butler - mdb815

## 1.1 System description

FBMS is an automated backup and revision program. The user specifies a folder that they want to keep backed up (the "live directory") and the location to store the backup (the "backup directory"). The program automatically searches the live directory for changes in intervals, and copies changes to the backup directory. Revisions are automatically created by creating diff ("patch") files for every change.

Thus, not only is the user's data backed up, but older versions of the data are also backed up. FBMS can be thought of as a hybrid of a local-only Dropbox and a version control system. While it's not as customizable as a version control system like SVN or git, FBMS is easy to use and runs in the background without the need for user interaction.

## 1.2 Business case

FBMS helps ensure the user's files are safely backed up without having to depend on limited cloud-based services or the complexity of programs like SVN. Programs like Dropbox keep revisions of files, but are limited in their ability to do so. FBMS removes these shackles, limiting you only by your hard drive capacity. FBMS currently supports multiple drive systems, including network drives, filling in the blanks of programs like Dropbox, which are online only.

FBMS differs itself from other backup systems in how it doesn't just backup files, but keeps revisions for safety. FBMS is targeted as the semi-casual audience, who know enough to understand the need for backups, but don't want to use a version control system (or perhaps don't like using such systems). The program is low effort to setup and requires no struggling with command line utilities. Once it's setup, it can run quietly in the background until you need it.

## 1.3 User-level goals for the system

**A simple to use backup solution for a user's files.**

- Ability to back up single files or whole directories.
- Granular backup for various restore points.
- User can specify any locally available drive to hold backup files.

**A file versioning and backup solution for their files.**

- User can recover deleted/lost files.

- User can revert to a prior version of a file.
- User can see a revision history of the file.

**Gives users a lightweight multi-platform software solution.**

- Able to use this same software on various operating systems due to Java architecture.
- Once setup requires little attention from user.
- Runs in the background creating a version history and back up of selected files.

## 1.4 User scenarios

**A user wants to backup files:**
The user will start up the software, databases and file repositories will be created or specified. The user will then go through the files/directories on the system and add them to the list of locations and files to be backed up. Once a location for the backup files and database are designed the software can grab copies and begin monitoring those files for changes. As changes are made, including name changes, the log will be updated and revisions created.

**User has had a critical system error and has lost all files:**
User can reinstall the software, specify the backup files and database location (e.g. an external drive) and restore everything that was specified for backup.

**User has accidentally deleted a file:**
The user can bring up the user interface for our program. They then select the file in question which brings up a list of prior revisions that have been stored. The user can then select the revision they desire to restore and the program will restore it to its directory.

**User has had a file change they don't want to keep or the file has corrupted:**
The user can bring up the interface. By selecting the file from the list first presented they will be shown a list of all revisions stored for the file. The user can select the version they wish to restore to and the current file in that directory will be reverted to that revision.

## 1.5 Scope document

- A front end for the user to specify files to be watched or restored.
- File monitoring system that makes notes of changes including creation, deletion and renaming.
- Database that stores file change history and relevant details about the file.
- Repository of diff files created and managed for our version control.
- The ability to maintain a backup of the latest revision to a designated drive.
- Ability to restore a file from a specified revision. This includes deleted or renamed files.
- The option to restore all files at a specified revision.

## 1.6 Project plan / Rough estimates

We broke the program into components and spread them out amongst the group members, trying to keep things fair and logical. We ended up with the following breakdown:

```
                                 Estimated time req
 MIKE:                                 (in hours)
     Class prototypes, PoD classes         3
     Watcher.*                             4
     Control.displayRevision()             3
     Control.displayRevisionChanges()      3
     Control.revertRevision()              4
     DbManager.init()                      3
     DbManager.getRevisionData()           3
     Data.getFolderContents()              3
                                          ---
                                  Total:  26

MICHAEL:
     Control.handleCreatedFiles()          4
     Control.handleModifiedFiles()         4
     Control.handleRenamedFiles()          4
     Control.HandleDeletedFiles()          4
     DbManager.renameFile()                3
     DbManager.getConfig()                 3
     DbManager.setConfig()                 3
                                          ---
                                  Total:  24

DA:
     FileHistory.getRevision()             3
     FileHistory.obtainRevision()          6
     FileHistory.storeRevision()           3
     FileOp.delete()                       3
     FileOp.filesize()                     3
     FileOp.rename()                       2
     FileOp.copy() (both)                  4
     FileOp.fileValid()                    3
                                          ---
                                  Total:  27

HATTAN:
     Control.restoreBackup()               4
     FileOp.createDiff()                   5
     FileOp.applyDiff()                    5
     FileOp.fileToList()                   3
     FileOp.isFolder()                     3
     Data.getRevisionInfo()                3
     DbManager.insertRevision()            3
                                          ---
                                  Total:  26
```

```
SYED:
     FrontEnd.*                            20
     FileHistory.renameRevision()           3
                                          ---
                                  Total:   23

--------------------------------------------
                          Grand total: 126
```

This does not include the collaboration time and putting the final program together. We'd estimate a total of 150 hours for the base project. If time allows, we have many other features that would be nice to implement.

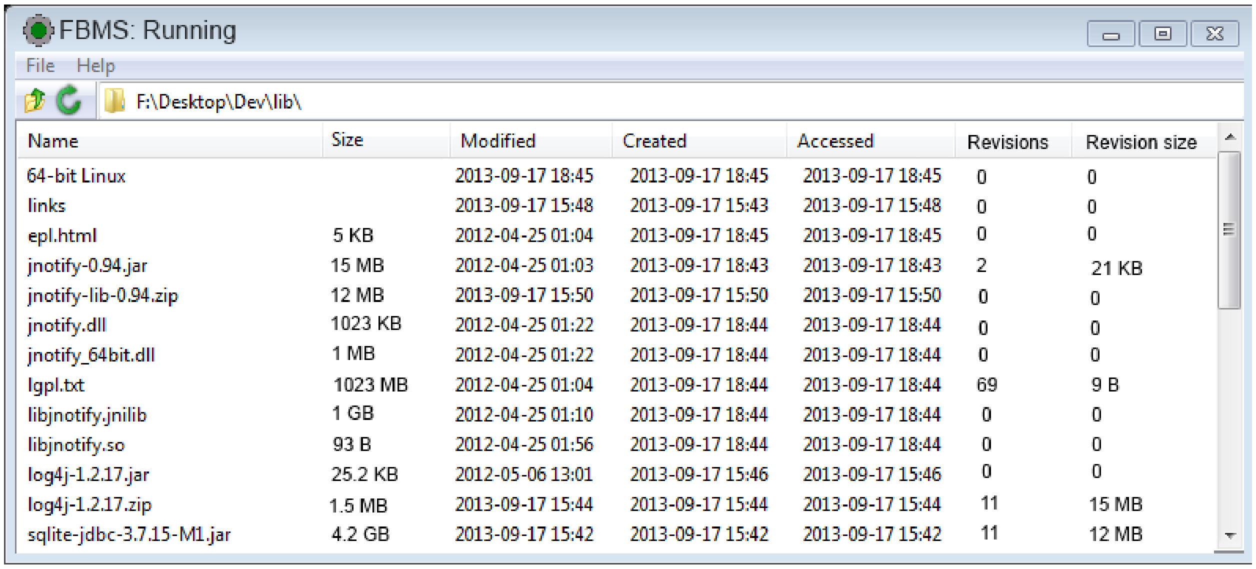
## 1.7 User involvement plan

We designed the system with rational uses in mind. Thus, we are not just the system's creators, but also the system's users. This eliminates the need for a dedicated user. Planning was done both individually and together, collaborating on ideas for both the system's use cases and interior workings.

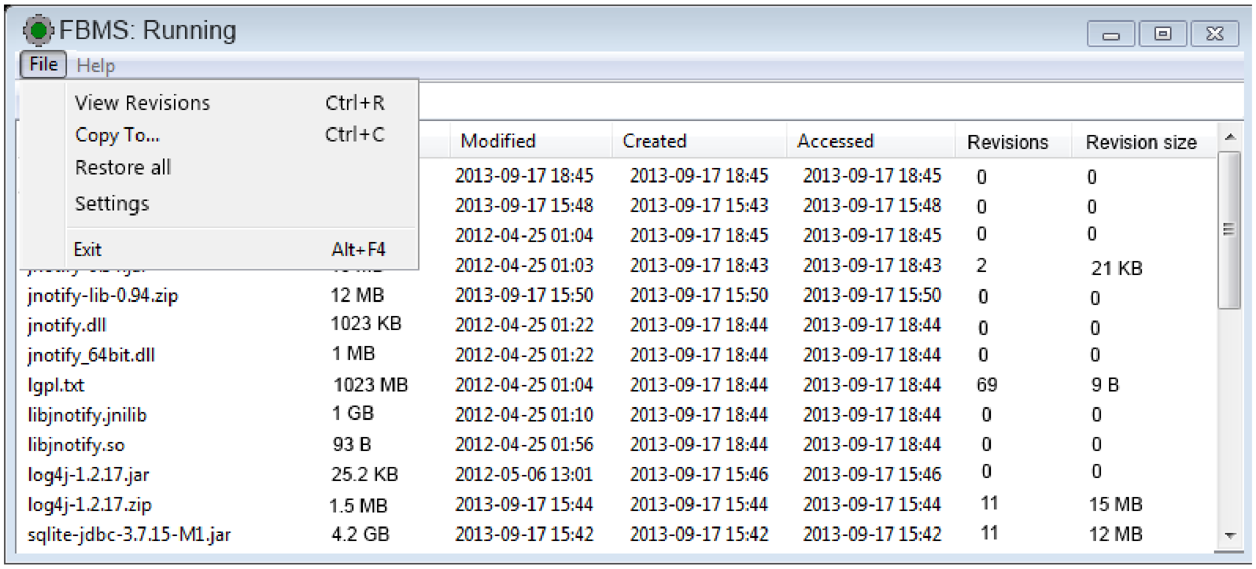
The time used from these users was heavily invested in the planning stage, going through several thousand words of planning. The users will continue to be used as we develop the program to ensure validity. We expect to ensure that the program follows the reasonably straightforward use-cases (which are very simple from the user's perspective).

## 1.8 Low fidelity prototypes

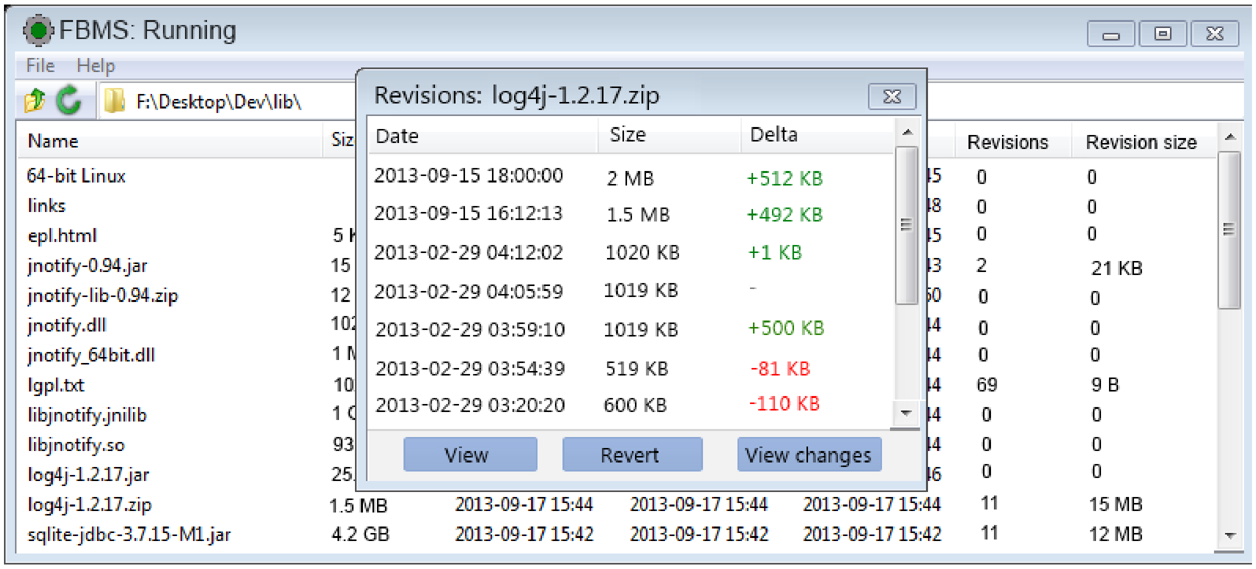
We designed some mock concepts on what the program would look like:



| Name | Size | Modified | Created | Accessed | Revisions | Revision size |
|---|---|---|---|---|---|---|
| 64-bit Linux | | 2013-09-17 18:45 | 2013-09-17 18:45 | 2013-09-17 18:45 | 0 | 0 |
| links | | 2013-09-17 15:48 | 2013-09-17 15:43 | 2013-09-17 15:48 | 0 | 0 |
| epl.html | 5 KB | 2012-04-25 01:04 | 2013-09-17 18:45 | 2013-09-17 18:45 | 0 | 0 |
| jnotify-0.94.jar | 15 MB | 2012-04-25 01:03 | 2013-09-17 18:43 | 2013-09-17 18:43 | 2 | 21 KB |
| jnotify-lib-0.94.zip | 12 MB | 2013-09-17 15:50 | 2013-09-17 15:50 | 2013-09-17 15:50 | 0 | 0 |
| jnotify.dll | 1023 KB | 2012-04-25 01:22 | 2013-09-17 18:44 | 2013-09-17 18:44 | 0 | 0 |
| jnotify_64bit.dll | 1 MB | 2012-04-25 01:22 | 2013-09-17 18:44 | 2013-09-17 18:44 | 0 | 0 |
| lgpl.txt | 1023 MB | 2012-04-25 01:04 | 2013-09-17 18:44 | 2013-09-17 18:44 | 69 | 9 B |
| libjnotify.jnilib | 1 GB | 2012-04-25 01:10 | 2013-09-17 18:44 | 2013-09-17 18:44 | 0 | 0 |
| libjnotify.so | 93 B | 2012-04-25 01:56 | 2013-09-17 18:44 | 2013-09-17 18:44 | 0 | 0 |
| log4j-1.2.17.jar | 25.2 KB | 2012-05-06 13:01 | 2013-09-17 15:46 | 2013-09-17 15:46 | 0 | 0 |
| log4j-1.2.17.zip | 1.5 MB | 2013-09-17 15:44 | 2013-09-17 15:44 | 2013-09-17 15:44 | 11 | 15 MB |
| sqlite-jdbc-3.7.15-M1.jar | 4.2 GB | 2013-09-17 15:42 | 2013-09-17 15:42 | 2013-09-17 15:42 | 11 | 12 MB |



| | | Modified | Created | Accessed | Revisions | Revision size |
|---|---|---|---|---|---|---|
| | | 2013-09-17 18:45 | 2013-09-17 18:45 | 2013-09-17 18:45 | 0 | 0 |
| | | 2013-09-17 15:48 | 2013-09-17 15:43 | 2013-09-17 15:48 | 0 | 0 |
| | | 2012-04-25 01:04 | 2013-09-17 18:45 | 2013-09-17 18:45 | 0 | 0 |
| | | 2012-04-25 01:03 | 2013-09-17 18:43 | 2013-09-17 18:43 | 2 | 21 KB |
| jnotify-lib-0.94.zip | 12 MB | 2013-09-17 15:50 | 2013-09-17 15:50 | 2013-09-17 15:50 | 0 | 0 |
| jnotify.dll | 1023 KB | 2012-04-25 01:22 | 2013-09-17 18:44 | 2013-09-17 18:44 | 0 | 0 |
| jnotify_64bit.dll | 1 MB | 2012-04-25 01:22 | 2013-09-17 18:44 | 2013-09-17 18:44 | 0 | 0 |
| lgpl.txt | 1023 MB | 2012-04-25 01:04 | 2013-09-17 18:44 | 2013-09-17 18:44 | 69 | 9 B |
| libjnotify.jnilib | 1 GB | 2012-04-25 01:10 | 2013-09-17 18:44 | 2013-09-17 18:44 | 0 | 0 |
| libjnotify.so | 93 B | 2012-04-25 01:56 | 2013-09-17 18:44 | 2013-09-17 18:44 | 0 | 0 |
| log4j-1.2.17.jar | 25.2 KB | 2012-05-06 13:01 | 2013-09-17 15:46 | 2013-09-17 15:46 | 0 | 0 |
| log4j-1.2.17.zip | 1.5 MB | 2013-09-17 15:44 | 2013-09-17 15:44 | 2013-09-17 15:44 | 11 | 15 MB |
| sqlite-jdbc-3.7.15-M1.jar | 4.2 GB | 2013-09-17 15:42 | 2013-09-17 15:42 | 2013-09-17 15:42 | 11 | 12 MB |



| Date | Size | Delta |
|---|---|---|
| 2013-09-15 18:00:00 | 2 MB | +512 KB |
| 2013-09-15 16:12:13 | 1.5 MB | +492 KB |
| 2013-02-29 04:12:02 | 1020 KB | +1 KB |
| 2013-02-29 04:05:59 | 1019 KB | - |
| 2013-02-29 03:59:10 | 1019 KB | +500 KB |
| 2013-02-29 03:54:39 | 519 KB | -81 KB |
| 2013-02-29 03:20:20 | 600 KB | -110 KB |

| Name | Siz | | | | Revisions | Revision size |
|---|---|---|---|---|---|---|
| 64-bit Linux | | | | | 0 | 0 |
| links | | | | | 0 | 0 |
| epl.html | 5 k | | | | 0 | 0 |
| jnotify-0.94.jar | 15 | | | | 2 | 21 KB |
| jnotify-lib-0.94.zip | 12 | | | | 0 | 0 |
| jnotify.dll | 102 | | | | 0 | 0 |
| jnotify_64bit.dll | 1 M | | | | 0 | 0 |
| lgpl.txt | 10 | | | | 69 | 9 B |
| libjnotify.jnilib | 1 G | | | | 0 | 0 |
| libjnotify.so | 93 | | | | 0 | 0 |
| log4j-1.2.17.jar | 25 | | | | 0 | 0 |
| log4j-1.2.17.zip | 1.5 MB | 2013-09-17 15:44 | 2013-09-17 15:44 | 2013-09-17 15:44 | 11 | 15 MB |
| sqlite-jdbc-3.7.15-M1.jar | 4.2 GB | 2013-09-17 15:42 | 2013-09-17 15:42 | 2013-09-17 15:42 | 11 | 12 MB |

Closing the window does not close the program, as most of the program works in the background. The window can be re-opened by clicking an icon in the user's system tray. This interface runs on a separate thread from the main program.

## 1.9 Project management report

The group has been meeting regularly after tutorials, and email correspondence has been heavy. There has been three meetings of approximately 10-20 minutes in length, so far, and we plan to continue this pattern until the project is complete. We did not keep a good track of time, but would estimate that we spent approximately five hours preparing for the design document that Milestone 2 was partially based on. We ultimately made a number of changes, which resulted in a complete re-write of the design document. This document represents an estimated ten hours of planning, thought, and creation. It experienced a number of rolling revisions and is the master document for the program's plan.

The master document may be viewed at <https://code.google.com/p/fbms/wiki/TechnicalDetails>. The changes that were made over time may be viewed at <https://code.google.com/p/fbms/source/browse/wiki/TechnicalDetails.wiki>. The documents were created by Mike Hoffert, but with the correspondence and planning of the entire group. They were supplemented with a number of diagrams created by Da Tao.

The project has several risks, particularly pertaining to "venturing into the unknown". The project requirements will work heavily with areas that most of the group members have no experience working in. It's entirely possible that the learning time for some of the members may cause the time estimates on the project to increase. The biggest risk, therefore, is the possibility of running out of time before everyone is able to complete their required components.

Another risk is keeping things cross-platform compatible. While our team spans different operating systems, we each individually use a single OS for the most part, which could lead to troubles if the different operating systems react to our code differently. We hope this will not be the case, and have taken every possible step to consider other platforms, including choosing libraries that are cross platform compatible and doing development on cross platform tools.

## 1.10 Project plan

The project plan largely corresponds to the TechnicalDetails document linked in 1.9 and the project plan in 1.6. The various components were largely designed to be able to work individually for a period of time, with unit testing to ensure the component works. Once the group members have constructed their components, we may begin integration testing, which will be done by the programmers of each individual module (as mentioned in 1.6).

With the components completed, the final integration is necessary, which will ensure the entire program works as expected. This will be done by Mike Hoffert, and then verified by the other group members. The time to do this will likely vary wildly, so we'll take the conservative estimate of eight hours (which assumes problems crop up). If severe problems crop up at this stage, it may be necessary to call an emergency meeting to focus on the integration.

With the programming working as expected, the distribution will be handled by Da Tao, who will package the program in a way that includes the appropriate licensing information (mandatory, as the libraries are all under either the Apache license or the LGPL). The distribution will ensure that the program can be easily run on all operating systems without the need to use the command line. This could take up to five hours, but will likely take less than an hour, since we created a demo program which can be adapted for this purpose.

## 1.11 Toy Example

As a toy example, we created a demo program that highlights the use of the libraries and major functionality: it detects file changes in the binary's folder, creates a database and reads from it, creates a logging file and writes to it, and creates a diff of two files, then restores one file from the other. We believe this is adequate to demonstrate that we understand the requirements of the project.

In fact, in the creation of the demo, we had to revise our technical document several times. For example, we originally planned to pass the result set from a database query to other components of the program, but that's neither safe nor easily done. Instead, it is necessary to copy the output of a database query to a list, which we can pass around freely.

The demo program was included in a zip file with this submission. Once extracted, it can be run by simply running `ant` in the folder with the`build.xml` file. The ant file will automatically run the program. Once ant has built the program, the program can be run standalone by running the`run` batch file or shell script in the same folder.

The program's files can also be downloaded from the SVN repository at <https://code.google.com/p/fbms/source/browse/#svn%2Ftrunk%2Futil%2Fdemo>.