Team: Anirudh Tiwari, Sneha Parmar, Saurabh Sood, Daniel Milroy

Title: NextBox - Manage your files with ease

Features that were implemented

<u>Use Case</u>	<u>Description</u>
UC 01	Create an Account
UC 02	Add Credit Card
UC 03	Rename File
UC 04	Delete file
UC 07	Create Directory
UC 08	Delete Directory
UC 09	Move Directory
UC 10	Rename Directory
UC 11	Upload File
UC 12	Download Files
UC 14	View Photo
UC 15	Search for file or directory
UC 16	See account usage
UC 17	Change Plan
UC 18	Share File
UC 19	Check Bill
UC 20	Create User
UC 21	Change activation status

UC 22	Create Plan
UC 23	Modify Plan

Features that were not implemented

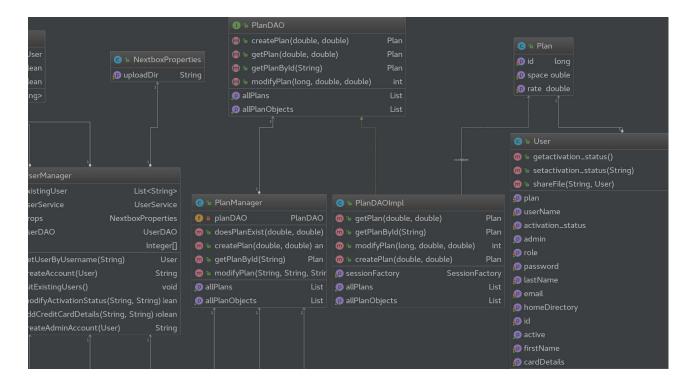
<u>Use Case</u>	<u>Description</u>
UC 05	Copy File
UC 06	Paste File
UC 13	View Text File
UC 24	Edit Text File

Class Diagram

We changed our class diagram a lot from our submission in Part 2. We discovered that files, directories, and filesystem paths could be more efficiently handled by an overarching class (Filepath) that would display and manipulate these objects. We realised that the interaction with the database is a perfect fit for the Proxy design pattern, which our Part 2 design did not accommodate. We added the Proxy classes and the required interfaces during the implementation. Also, we ended up removing a lot of classes for the use cases we could not implement (Such as the Clipboard class, which would have been used for the Copy-Paste file use case). See page 6 for our Part 2 class diagram and our final class diagram on page 7.

Design Patterns

Our project made heavy use of the Proxy design pattern, which we used to interact with the Database.



In the above class diagram, we have a proxy class called **PlanManager**, an interface called PlanDAO, and a concrete class called **PlanDAOImpl**, which implements **PlanDAO**. The proxy class comprises of a **PlanDAOImpl** object, and can use it to call methods like **getPlan**, which create or delete Plan objects from the database.

What we learned

The overall design process really helped us plan out the project well. The diagrams helped us to avoid potential problems during the development, and made development easier and faster. The Activity Diagrams really helped us flesh out what the User Interaction will be for a particular

Use Case. Moreover, the Sequence Diagram really helped us in the coding process, as we knew beforehand what classes we need to create, and what methods we need to implement. It really helped speed up the coding process.

We had a great learning curve with using Java and Spring MVC and we learnt how to think in terms of the MVC architecture. It greatly simplified organization of code for a web application like ours

We also learnt how to think from different perspectives for a software product - starting with thinking like a customer and a solution or program manager and progressing more towards the view of a software developer. It was interesting to experience how they could have differences of opinion in terms of scope in the real world, and to realize how it would make things easier when everyone is on the same page right from the start.

More generally, we learned how to work as a team to build a cohesive software product in GitHub. The organizational logistics necessary to coordinate simultaneous code revisions and changes can be difficult even for a semester project. We learned to collaboratively review pull requests, as well as resolve complex merge conflicts when pulling in new features.

Code Usage References

1. Hibernate:

http://www.roseindia.net/spring/spring4/login-form-using-spring-mvcand-hibernate.shtml

- Search file: (line 20-32 in FilesystemService.java)
 http://stackoverflow.com/questions/15624226/java-search-for-files-in-a-directory
- Download file (lines 165-182 in FilesystemAPI.java):
 http://howtodoinjava.com/spring/spring-mvc/spring-mvc-download-file-controller-example/
- Exception handling for delete/move/rename files and directories (see corresponding methods in FilesystemAPI.java): https://docs.oracle.com/javase/tutorial/essential/io/delete.html
- Checking MIMEtype for photo (lines 54,56 in Filepath.java):
 http://stackoverflow.com/questions/9643228/test-if-file-is-an-image
- Displaying photo in JFrame (lines 193-207 in FilesystemAPI.java):
 http://stackoverflow.com/questions/14353302/displaying-image-in-java

