Sprint 2 Design Document

BYB Team

Purpose

The purpose of our application in this sprint is to create a server-client infrastructure that allows a remote program to view, edit, and save a business plan stored in a centralized server, using the implementation of BusinessPlan class created in the previous sprint phase. We focus on developing a basic functional program to allow remote communication without considering the production-level quality and security of software.

Specifications

On the server side, all people have a username, a password and a department they belong to. People can only belong to one department. Some people are also administrators. Administrators can add other people to the system along with their departments and their passwords to the system. Each business plan also has a year, along with a department that it is associated with. Each business plan has a boolean to indicate if it is editable by non-administrators and every 2 minutes the server saves all the data in memory to file(s) on the disk. When the server starts, it reads the files on the disk.

On the client side, when client programs startup they require a person to submit their username and password for authentication. Once authenticated, the client will submit actions on behalf of that person. Clients may retrieve any business plan for their user's department, based on year. If a given business plan does not exist, a suitable message will be sent. Clients may make changes to a business plan to a local copy and then and resubmit their copy to the server to replace the version on the server, but only if the business plan is still editable. Clients may generate a new business plan either as a "blank" document or as a copy of the department's business plan of a different year. If the client's user is an administrator, they may also add persons, along with passwords, and departments. An administrator may turn a particular business plan on or off for editing. All clients in a department have permission to view, edit and save business plans for their department.

Design Overview

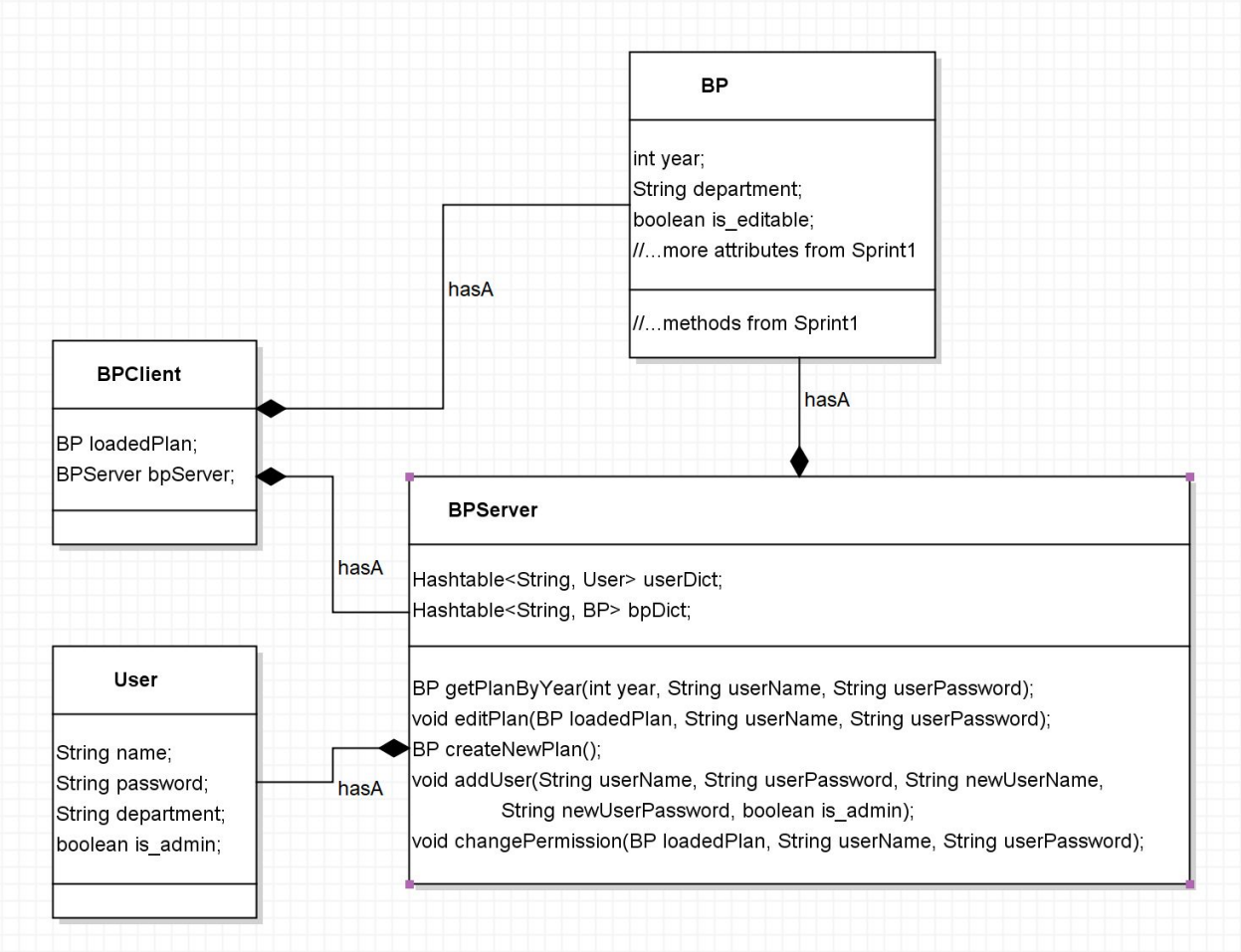

Figure1: Class Diagram

Figure1 is our class diagram for Server-Client Design. Overall, it is a proxy pattern that provides a proxy for business plan to control access to it. It allows clients to do something with BP but also protects BP from being directly touched by clients.

BPServer is used to store all users' information and business plans by using dictionaries and to handle all the requests from BPClient. Therefore, there are two attributes in the BPServer - *userDict* and *bpDict*. A user object can be accessed by user name, since we assume that no two users have the same name. Similarly, a business plan object can be found by the concatenation of year and department. There are five methods that the BPClient can use by first connecting to the BPServer using RMI Registry. Users can get their department's business plan in a specific year (*getPlanByYear()*). They need to type in their username and password for the

BPServer to authenticate. If the information given by a user does not match that in the userDict, an error message will be sent back to the user. A BP object will be returned to the BPClient for users to view or to do other actions. The BP object will be stored in the BPClient as *loadedPlan*. We wanted to have the BPServer return a XML file of the object before, but then we thought it should be better to have an object for easier editing. Users can edit a business plan if it is editable (*editPlan()*). They can also create a new business plan and an empty object will be given (*createNewPlan()*). While some users are administrators, they can also add other users into our system (userDict) by typing in their information (to verify whether they are admin) and a new user's information, including a username and a user password (*addUser()*). Also, as administrators, they can turn a business plan on or off for editing (*changePermission()*). One more thing we think we can do in the future is to have a login at the very beginning so we do not have to verify users' identify each time they take an action.

BPClient is a very simple client class that has connection with BPServer by RMI. It only has memory for the current BP (*loadedPlan*) that the user is viewing or editing. And it has a *BPServer object* to do everything for it.

User represents people who are going to use our system. They have an unique *name* and a *password*. They belong to a *department* and they can be an administrator or not (*is_admin*).

BP is what we have from the last Sprint. Few fields are added here as required. Each business plan has a *year* and a *department* that is associated with. Also, business plans can be turned on or off for editing by administrators.

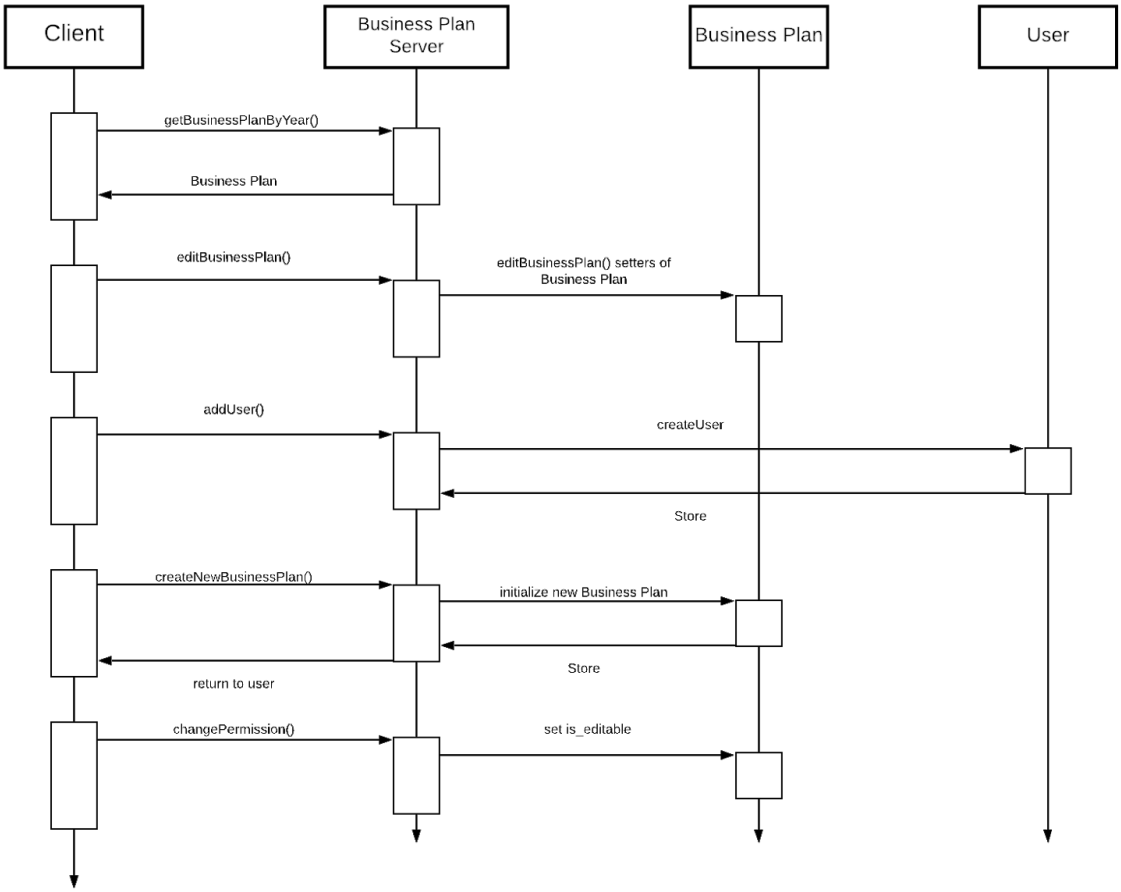

Figure2: Sequence Diagram

Figure2 is a sequence diagram of how these different classes interact with each other.

- Since all the business plans will be saved in the Server, when we want to get a business plan the client only needs to interact with the Server to get the business plan it wants.
- When we try to edit a business plan, we call the editBusinessPlan method of the Server and then the Server will call the setters of the BusinessPlan to edit business plans.
- When we call addUser, the Server's addUser method will be invoked. Inside that method, a new user will be initialized. After initializing the new user, it will be returned to the Server and saved there in the user dictionary.
- The createNewBusinessPlan method is invoked when the user wants a new blank business plan. For this method, the client calls the Server's createNewBusinessPlan method, a new instance of Business Plan will be initialized, returned and stored in the

Server's Business Plan dictionary. Finally, the new Business Plan will be given to the client for the user to use.

- The administrator can use changePermission method to change the editability of a business plan. The client calls the user's changePermission and lets the Server do actual change which is just calling the setter of the is_editable attribute of the business plan. This method will not return anything back to the client just like the editBusinessPlan.