PENNSTATE Final Report

Department of Electrical Engineering and Computer Science

DCG Software Value Agile Competency Development

May 2, 2016

Janice Huang
David Reidler
Sergey Ivanov
Dylan Fetch
Archan Shah

At the bottom left of title page, be sure to indicate whether or not IP and NDA apply:

☐ or ☑ Intellectual Property Rights Agreement Applies

Executive Summary

Agile methodology has made a large impact in software development, and as it stands, most companies have already made the philosophical shift to that framework. However, what's currently missing from agile is a well-structured incentive system. DCG proposes to create an evaluation and training framework centered around distinct role and skill combinations in order to help HR quantify and reward good performance.

Table of Contents

1.0 Introduction

1.1 Initial Problem Statement

While agile is a good methodology for organizing employee division of labor, it doesn't provide nearly as strong a framework for asking for career advancement and pay-raises. Such a system is impractical in the real world and requires both tools and guidance to help impartially identify the employee's contributions and skills.

1.2 Objectives

In order to solve the problem, the project created by DCG intends to create a fully fledged management solution that aims in providing a comprehensive, integrated HR toolbox for managers and teamleaders to both identify problems and reward performance.

2.0 Customer Needs Assessment

2.1 Gathering Customer Input

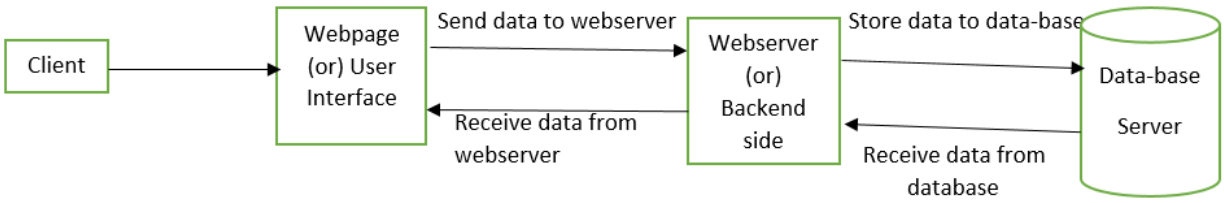
The customer for this project is our client, DCG Software Value. The initial customer needs were outlined in the project description given to us at the start of the semester. We refined these needs in a meeting with our client. We continue to refine and add needs when we meet with our client every other Thursday. That is because we are using the Agile software development lifecycle and only plan two weeks in advance. Our client's desires may change from sprint to sprint based on what they see in the product and research they do on their own time.

2.2 Weighting of Customer Needs

The customer would like a website that is able to import BSMImpact individual assessments, have role profiles and job descriptions mapping to assessment results, import BSMImpact organizational assessments, have individual and organizational candidate reports, a submenu for creating personnel accounts, a submenu for an HR admin, a submenu for individual assessments, a submenu with information about the DCG ACD framework, a submenu for roles and job descriptions, HR personnel account access, admin account access, manager account access, and employee account access.

3.0 Concept Generation

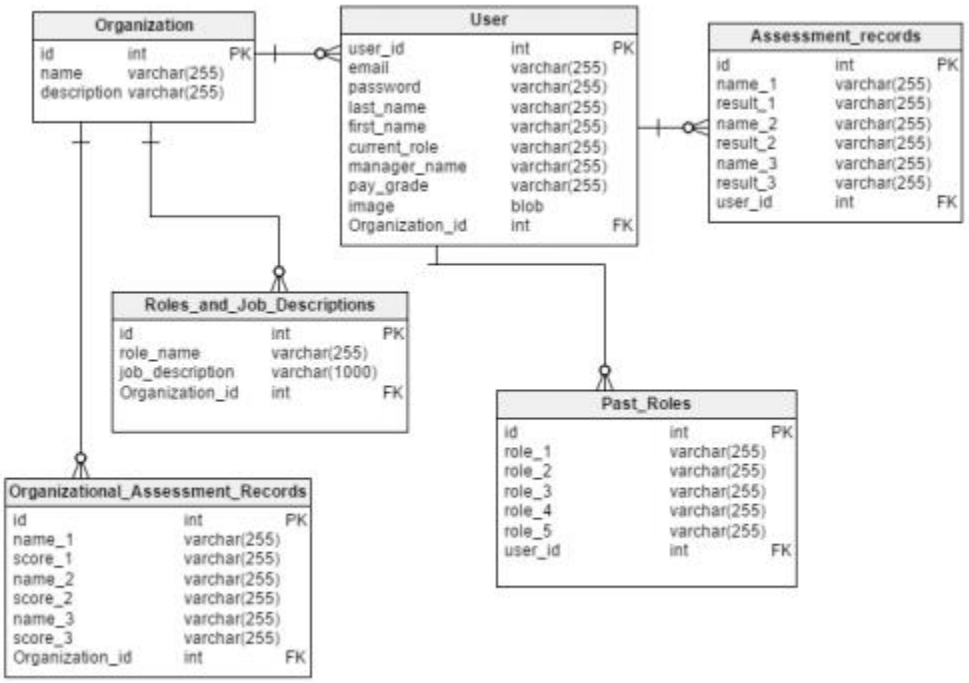
3.1 Problem Clarification



3.2 Concept Generation

1. The client (employee) needs to login first. Once client is logged in, check his current role as employee.
2. If the current role is HR, then he/she has access to all the data. They are admins.
3. If the current role is manager, he/she has read only access to his/her data and employee data who all work under him/her.
4. If the current role is employee, he/she has read only access to his/her data.

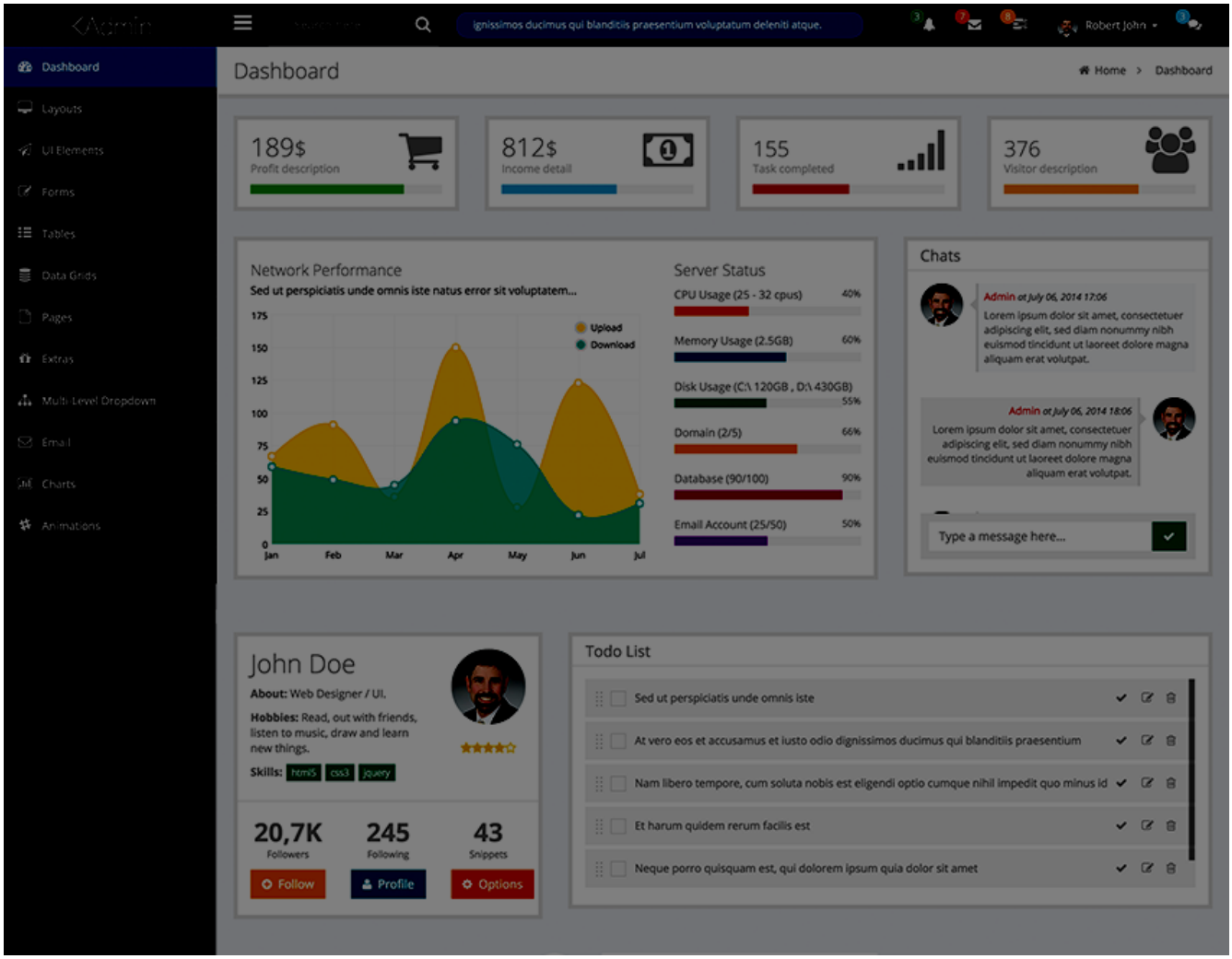
5. All employees has to take Individual Assessments to know their skills level, and all the records will be stored in to database under the employee's record.
6. There will be a design for how to analyse individual assessments, but we haven't planed it yet. As we only do planning for 2 weeks under agile scrum methodology.
7. Data-base design that we have now is given below, which is subject to change.



4.0 System Level Design

For this project our team will be utilizing two virtual private servers to host a test website and a production website, both of which are being hosted on a LAMP stack. That is, the operating system for the server will be Linux, specifically Ubuntu Server. The web pages will be served by the Apache web-server, and configured to use a self-signed SSL certificate for HTTPS connections. The database choice for the website is MySQL, and PHP will be used to interface between the web interface, and the database. To assist with maintaining the database, phpMyAdmin will be used to provide direct access to the MySQL server without the need of SQL statements. SSH access has been provided and will be used to perform any necessary system maintenance or modifications.

For the front-end, our team will be using a Bootstrap theme called KAdmin (https://shapebootstrap.net/item/1524954-kadmin-free-responsive-admin-dashboard-template) to help create eye appealing forms and pages. Using this theme has the advantage of allowing us to easily take advantage of Bootstrap without using the bland defaults. An example of the Bootstrap template, KAdmin, is shown below.



The rest will be written in pure HTML and any additional JavaScript that is necessary.

As for the back-end, our team has designed a database architecture that will be able to store all necessary, persistent data on the site. We will be using phpMyAdmin to make manual adjustments as necessary, but all accesses to the database will specifically be done through our own PHP code to give us full control and responsibility over the database storage. In addition, our team will need to sanitize all data sent over from the website to the back-end.

5.0 Special Topics

5.1 Budget and Vendor Purchase Information

The budget for this project will largely remain untouched. The only expense that is required for this project is the mandatory sixty-two dollars for our poster for the showcase. DCG Software Value have supplied us with all services that we will need, and anything else will be from an open source.

5.2 Project Management

For this project, we are implementing the Scrum methodology of the agile model. In this method, we only have goals for the two weeks ahead. At the end of two weeks, we meet with DCG Software Value to discuss what we have done and to plan for the next two weeks. In addition to the bi-weekly meetings, each of us will take turns at being the Scrum Master, who is responsible for the weekly report and contacting our sponsors as necessary. Below is the agreed upon dates for our sprints.

Sprint Phase	Sprint Review	Scrum Master
First meeting	January 26, 2016	Dylan
1	February 12, 2016	Janice
2	February 25, 2016	David
3	March 17, 2016	Archan
4	March 31, 2016	Sergey
5	April 14, 2016	Dylan

We used the online tool, *Trello*, to keep track of what tasks still needs to be done, in progress, or finished

5.3 Communication and Coordination with Sponsor

We have a telecommunication with our client on Thursday every two weeks to discuss what we have done and what our new goals should be. In addition, we have created a Gmail to centralize all emails to and from DCG Software Value.

6.0 Detailed Design

6.1 Component and Component Selection Process

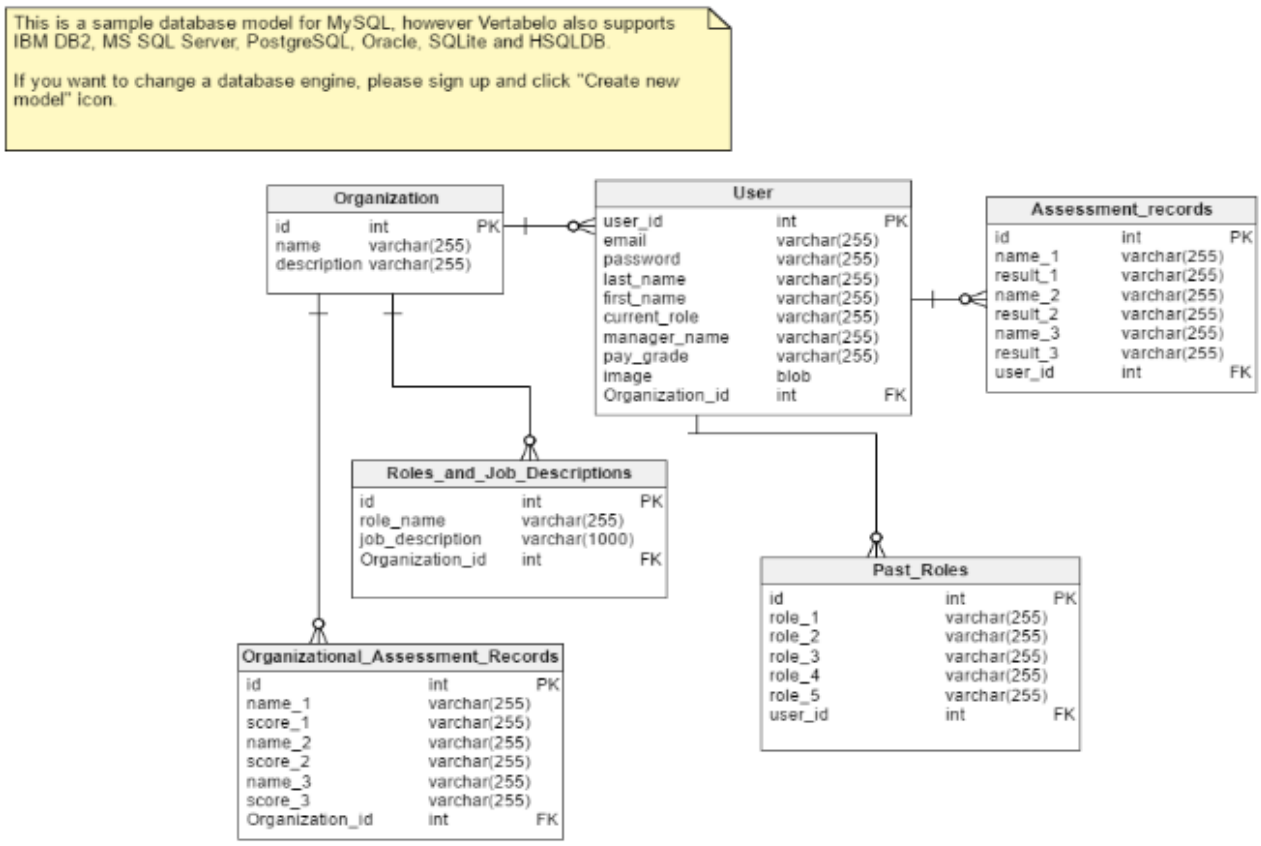
During the design phase, we were faced with various choices of software to integrate into our project. To choose these, we needed to draft up the project requirements and determine what features we needed. For our project, we needed a database to store specific application data, such as login credentials, employee information, etc, we also needed a web server to host and serve the website we were building. This also comes with the implication that the machine we need should also be able to run all of these applications, and allow us to manage them ourselves. In summary, we needed some kind of machine running some kind of DBMS and some type of web server that was able to run PHP.

Our final decision was to go with an Amazon AWS server running Ubuntu 12.04, and run a LAMP stack on the server. A LAMP stack includes a PHP capable web server (apache) and MySQL (DBMS), which is exactly what we needed for the project. A LAMP stack is a very reliable and well-tested method of hosting database driven applications, and is very configurable to the very last detail. This ensured a solid foundation that our project will be built upon, and allows ample flexibility for development. A LAMP stack also has the benefit of being a free solution, meaning that not only is it well tested, but it is also very affordable from an economic standpoint since everything in the LAMP stack is free software.

We had considered other alternatives such as a Node.JS server, or nginx, but these had some qualities which made them a much less attractive solution. Specifically, we found Node.JS to be much more complicated to implement a simple web server. It turns out that we need to design and implement our own web server in JavaScript, which is something we didn't feel was necessary for our project, and would hinder our ability to develop in such a short timeline. Nginx turned out to be much more complicated, and didn't quite have the maturity of Apache. It seemed as though apache was much stable, which is something that the project needed to be transitioned into a production environment. We also considered IIS by Microsoft, but setting up, configuring, and managing IIS would require a much larger time investment for no apparent gain. Finding a suitable host for IIS was much more expensive as Windows Servers were much more expensive than a Linux server, and with no apparent gain, it just didn't have enough benefits to be a viable solution.

6.2 Database Design

For this project, we are using MySQL for the database. Below is the list of tables, their attributes, and a description of what each attribute holds.



- Organization: It holds data for different organizations.
  - Id: It holds organization id, which is an integer and primary key.
  - Name: It holds organization name, which is a string.
  - Description: It holds description for organization, which is a string.

- Roles_and_Job_Descriptions: It holds different role names and job descriptions for those role names.
  - Id: It holds id for this table. Which is an integer and primary key.
  - Role_name: It holds role names, which is a string.
  - Job_description: It holds job descriptions, which is a string.
  - Organization_id: It is foreign key that relates this table to Organization table.

- Organization_Assessment_Records: It holds assessment records for different organizations.
  - Name_1, score_1, name_2, score_2, name_3, score_3: It holds assessment names and scores, which are all strings.
  - Organization_id: It is foreign key that relates this table to Organization table.

- User: It holds user accounts for organization.
  - User_id: It holds user id, which is integer and primary key.
  - Email: It holds user email address, which is a string.
  - Password: It holds password for user, which is a string.
  - Last_name: It holds last name for user, which is a string.
  - First_name: It holds first name for user, which is a string.
  - Current_role: It holds current role for user, which is a string.
  - Manager_name: It holds user's manager name, if user has one. It is a string.
  - Pay_grade: It holds pay grade for user, which is a string.
  - Image: It holds user image, which is a blob format.
  - Organization_id: It is foreign key that relates this table to Organization table.

- Past_Roles: It holds all past roles for user.
    - Id: It holds id for past roles, which is an integer and primary key.
    - Role_1, role_2, role_3, role_4, role_5: All holds past role names for user, which are all strings.
    - User_id: It is foreign key that relates this table to User table.

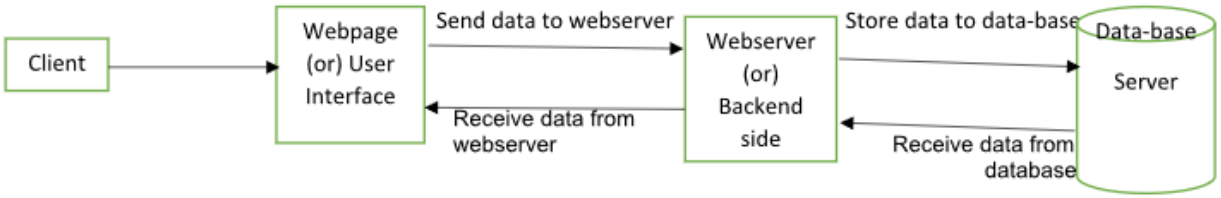
- Assessment_records: It holds assessment records for user.
    - Id: It holds id for past roles, which is an integer and primary key.
    - Name_1, result_1, name_2, result_2, name_3, result_3: All holds assessment names and their results for user.
    - User_id: It is foreign key that relates this table to User table.

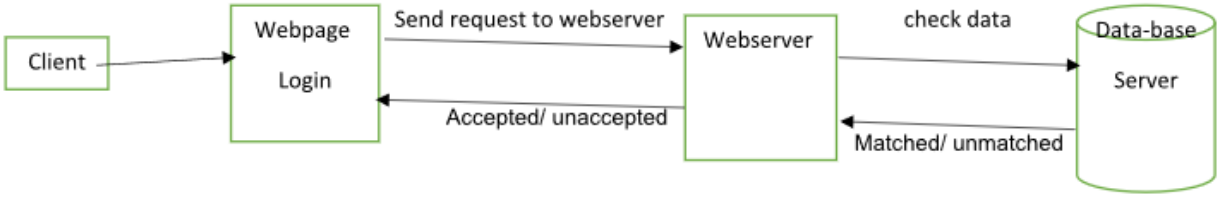
6.3 Use Case Definition

When designing the test cases, we were told by our sponsors what types of employees would be using the website. We then imagined ourselves in those roles, and considered what we should be allowed to do if we were that type of employee. In accordance to the SCRUM methodology, we came up with user stories that suited each type of user. Listed below are the stories that we came up with. We determined the story points by how important they are to be completed.

- As an employee I want to access my own account with my pay grade, hours of work so that I can keep track of my work status.
  User story point = 2
- As an HR personnel I want to access all accounts so that I can see and update their information and position in the company.
  User story point = 2
- As an admin I want to be able to access all accounts so that I can delete and reactivate accounts in case of retired employees or an employee returning to the company.
  User story point = 1
- As a manager I want to access all accounts so I can determine the position the employees of my department
  User story point = 2

6.4 Interface Design





There will be three types of inputs:

- Store data to database
- Get data back from database

- The login request

6.5 Test Procedures

The front end is going to have various menus, forms, and links. In order to test the menus and links, we simply click them and see if they work. In order to test the forms, we will submit input to them and check the database to see that if they were added, updated, or deleted. We will try both inputs that should and shouldn't work. We will make sure that the inputs that are supposed to work, work as expected, and that the inputs that are not supposed to work (i.e. don't fit the format of the form) do not work. The back end will have API calls that interact with a database. In order to test them we will submit inputs that we expect to work and, depending on the nature of the API call, check to see if the returned value is correct or that the correct value is stored in the database. We will also submit inputs that we do not expect to work and make sure that the error is handled correctly and the incorrect data is not submitted to the database. In order to test that our production server is working, we can simply visit the site and check it. All that we need to check is that everything is displayed properly; the functionality will already be tested.

7.0 Final Discussion

7.1 Construction Process

After deciding on using raw PHP, Javascript and mySQL, we created our own template based on the KAdmin front-end template from Bootstrap, and proceeded to add features to the site. After procuring an amazon web service instance to push our website to, we created the static pages: home, admin, assessments, login, resources, and about. We then created the admin panel to verify our format for the database and our submitting JSON objects containing our specified user format. After this, we created more in-depth features with the Create_user, Delete_user, JobDesc, Update_user, HTML pages to perform their respective database operations and debugged and verified those as well.

7.2 Test Results and Discussion

The test process mostly consisted of refinement to the feedback we received from the customer. This is largely qualitative and focused on high-level functionality and appearance of our page. After our initial version of the site that just submitted users, we were notified to change the theme from a mostly dark website to blue, red and white to fit the DCG Software Value theme as well as integrate their imagery and icons. As far as functionality is concerned, we resolved how the forms were to send the information from the front-end to the back-end. We did run in some trouble with pushing into an array, such as the assessment and roles field. However, we were able to debug and fix why those information were not being pushed correctly into our database.

8.0 Conclusion and Recommendations

Looking back, the process was largely one about discovering what image the website should portray and how to structure our functionality. While using PHP was a solid, well-established procedure in creating a site, we would have benefitted more from a framework language, such as ruby on rails or Django. Most of our development problems came from small, insignificant misunderstandings between the disparate front-end and back-end and a framework language effectively just takes a template of a website and constructs both, only requiring the interface to be specified from the coders.

Another major issue in development was the decision change from a blob-based storage of PDFs to a hierarchical structure. Unfortunately, this change was very last-minute and forced us to scrap our PHP. If solved much sooner, knowing we would use them, we could easily have avoided the mix-up.

For future developments, I recommend improving PDF storage by parsing the PDF, probably by using an existing php library, and storing only the necessary components. I would suggest focusing on user feel at this point and creating a nice interface for each type of perceived user, this would require presenting the stored DB info in the best format DCG would feel is appropriate.