Kyrne Beauford, Charlie Boggus, Kyle Kempf, Carson Susich, Sean Walston

AMIS 3610

April 15th, 2019

**PDFBox**

**Purpose:**

Too often, in business and personal ventures, individuals need to convert a document to PDF and are met and are without a reliable, safe way to do so. For many, the options are: pay for an expensive Adobe product or use a website that is sure to infect your computer with a slew of malicious malware. However, with the advent of PDF Box (patent pending) there is another way to convert common business documents without the malware and without the subscription. This product will fulfill the need of converting word, powerpoint, and excel documents to PDFs that many individuals and businesses run into every day.

**Architecture:**

For this project we decided to use Angular for the web frontend, an ASP.NET Web API for the API layer, and a PostgreSQL database for the database layer. The only dependencies being used in the system are in the API layer and they are as follows: a library to interface with the PostgreSQL database from ASP.NET, a JWT Authentication library for generating user authentication tokens, and the Microsoft Office Interop Word, Excel, and PowerPoint libraries for converting Microsoft Office documents to PDF format.

The database only has two tables: the Users table which stores user account information, such as their email, username, their password encrypted with HMAC SHA512 encryption, and their registration date. The second table is the Documents table which stores user-uploaded documents, and has attributes such as a foreign key reference to the user who owns the document, the document name, the document size, the actual document data, and the date on which the document was uploaded.

The API layer consists of three different controllers that control its functionality and interact with both the client-side frontend and the database: the Contact Form Controller, the User Controller, and the Document Controller.

The Contact Form Controller only has one HTTP POST method: Submit, which is a method that is called when a contact form is submitted, and receives contact form details (sender name, email, and message) from the HTTP request body, then proceeds to use an SMTP client to send the message as an email to the designated inbox.

The User Controller has six different methods, the most important of which are RegisterUser, AuthenticateUser, GetUser, UpdateUser, and DeleteUser. RegisterUser is pretty self-explanatory in that it receives new user details from the HTTP POST request body and then creates a new user from those details in the database. AuthenticateUser is another HTTP POST method that receives user login details from the HTTP request body and tries to find a user (and authenticate them) using the provided login details, then returns a generated authentication token back to the client. GetUser simply retrieves the account details (email, username, registration date) from the currently authorized user. UpdateUser, again, is relatively self-explanatory; this method takes user account information from the HTTP request body (new email, new username, or new password), and using those details, changes the account information of the currently authorized user in the database. Lastly, DeleteUser is simply a method to delete the account of the currently authorized user as well as any documents they have stored in the Document database.

The Document Controller has seven public HTTP methods, and a number of private utility methods for use in document operations, the most important of which are: DownloadDocument, UploadDocument, ConvertDocument, as well as the private utility methods to convert the different Microsoft Office file types to PDF format. DownloadDocument is an HTTP GET method that retrieves a Document Id number from the HTTP request route, tries to find a Document with the provided Id owned by the currently authorized user in the Document table, and if one is found, returns a File Blob back to the client for download. UploadDocument is an HTTP POST method that receives one or many files from the HTTP request body, as well as a flag for whether or not to convert those uploaded documents to PDF, then stores the uploaded files in the Document database and associates them with the currently authorized user’s account. ConvertDocument is much like the UploadDocument method, except that it does not require a user to be logged in to use, and does not store any documents or data in the database. It simply converts the uploaded document and returns a File Blob back to the client for download. The private utlity methods are simply for converting each of the different Microsoft Office file types to PDF using the Microsoft Office Interop libraries.

The client-side web frontend layer is made using Angular, and is split into three separate parts: the Account Module, the Home Module, and shared components, services, and models.

The Account Module only has three components, each of which can only be accessed if a user is currently logged in: the User Dashboard, the Documents page, and the Upload Documents page. The User Dashboard is where the user can change or update their account details, and this component calls the UpdateUser API method. The Documents page is where a user can view, download, or delete the documents they have stored in the database. This component utilizes the GetDocument, DownloadDocument, and DeleteDocument methods from the API DocumentController. The last component is the Upload Documents component. This component is where a logged in user can upload and store documents in the database, and this component utilizes the UploadDocument method in the API DocumentController.

The Home Module has a number of components, all of which are public and do not require a user to be logged in to access. The first is the About Us component, which is simply a page that outlines what PDFBox is. The next is the Contact component, which is simply a contact form that, when submitted, makes use of the Submit method in the API ContactFormController. Next is the Convert component which is a public Microsoft Office to PDF converter which allows users to upload a document, have it converted to PDF, and then download the converted document. Next is the FAQ component which is simply a page that shows frequently asked questions. The next component is the Privacy component which is simply a page that outlines PDFBox’s privacy policy. Lastly is the Index component, which is simply the homepage of the website.

The shared portion of the web frontend is mainly comprised of services, but has a few components that are shared site-wide. The first of these shared components is the global navigation bar at the top of the webpage, as well as the global footer that appears at the bottom of the webpage. The NotFound component is also located in the shared portion, which is simply a page that is displayed when an HTTP 404 Not Found error is received. The services that are shared are: an authentication guard service, which makes sure the user’s login token is valid as well as restricts access to pages if it’s not; an alert service, which is used to display alerts (such as success or error messages) across the webpage; and an authentication service which is used to interface with the API and register or authenticate a user using provided account details.

**Outcomes:**

There are several metrics we will use to measure the success of PDF Box. Metrics such as monthly users and documents converted will be a measure of how often the service is being used, letting us know how effective the service is. If we can maintain a steadily increasing user base of about one-hundred users a month we will consider the product to be successfully growing. However, if we are not able to meet those requirements for six months we will pivot are strategy possibly increasing advertising or the features of the service. If we are unable to maintain growth for a year we will consider folding on the project.

Because PDF Box will operate using a freemium revenue model, we will also be tracking the clicks our ads will be receiving to measure their effectiveness. We will aim to have all ad real estate sold within the first three months of operation and we will be selling ads directly to media buyers to maximize revenue. A premium account will be optional but offer users additional storage space for a monthly subscription. All users will have 10GB for free and be able to upgrade to 100GB for one dollar a month and 200GB for two dollars a month.

Along with tracking subscriptions we will be using general site metrics to keep an eye on the health of PDF Box. Website traffic, to monitor growth and exposure, traffic sources, to see referral vs direct traffic, and bounce rate. These metrics will be key in guiding us to our goal of becoming the number one PDF conversion utility on the web.