Design and Implementation Report

Project: Online Spreadsheet

Siddharth Bulia : 130050012 **Professor: N.L Sarda**

Rawal Khirodkar: 130050014

Lokit Kumar Paras: 130050047

A] Introduction:

The project focuses on designing an user friendly interface to create and maintain documents online. We are using Django-Python as our web platform and Sqlite for hosting the database schema needed for the project.

Following are the features implemented in the project:

- **User Profile creation**
- **Creation of multiple documents (each document can consist of multiple spread-sheets)**
- **Hosting Images and Videos in the spread-sheets**
- **User Document Privacy**: The owner can choose who can edit/view his/her documents.
- **Sharing Documents across Users**

Let us explore the database design in more detail.

B] Relational Model:

Entity sets:

1) **Users (Username, Password, Full Name, E-mail)**

   -consists of all the users possessing an account on our portal

2) **Documents (Doc Id, Doc Name, Created On, Created by)**

   -contains all the available documents

3) **Sheets (Sheet ID, Sheet Name, Total Columns, Total Row,Sheet Last Modified, Sheet Created on)**

   -contains all the sheets present on the page

4) **Cell (Cell ID, Cell X, Cell Y, Cell Color)**

-represents the cells present in various sheets.

5) **Data (Data ID, Data Size, Data Type)**

-contains the data that is stored in the cells of various sheets.

6) **Image (Data ID, Image Data, Image Title)**

-consists of the data which is of image type

7) **Text (Data ID, Text Data, Text Color, Font, Font Size)**

-consists of the data which is of text type

8) **Video (Data ID, Video Data, Video Title)**

-consists of the data which is of the Video type

Relationship Sets:

1) **Ownership (Username, Doc ID, Rights)**

-binary relation which relates entities User and Documents

2) **Contained In (DocID, Sheet ID)**

-binary relation which relates entities Documents and Sheets

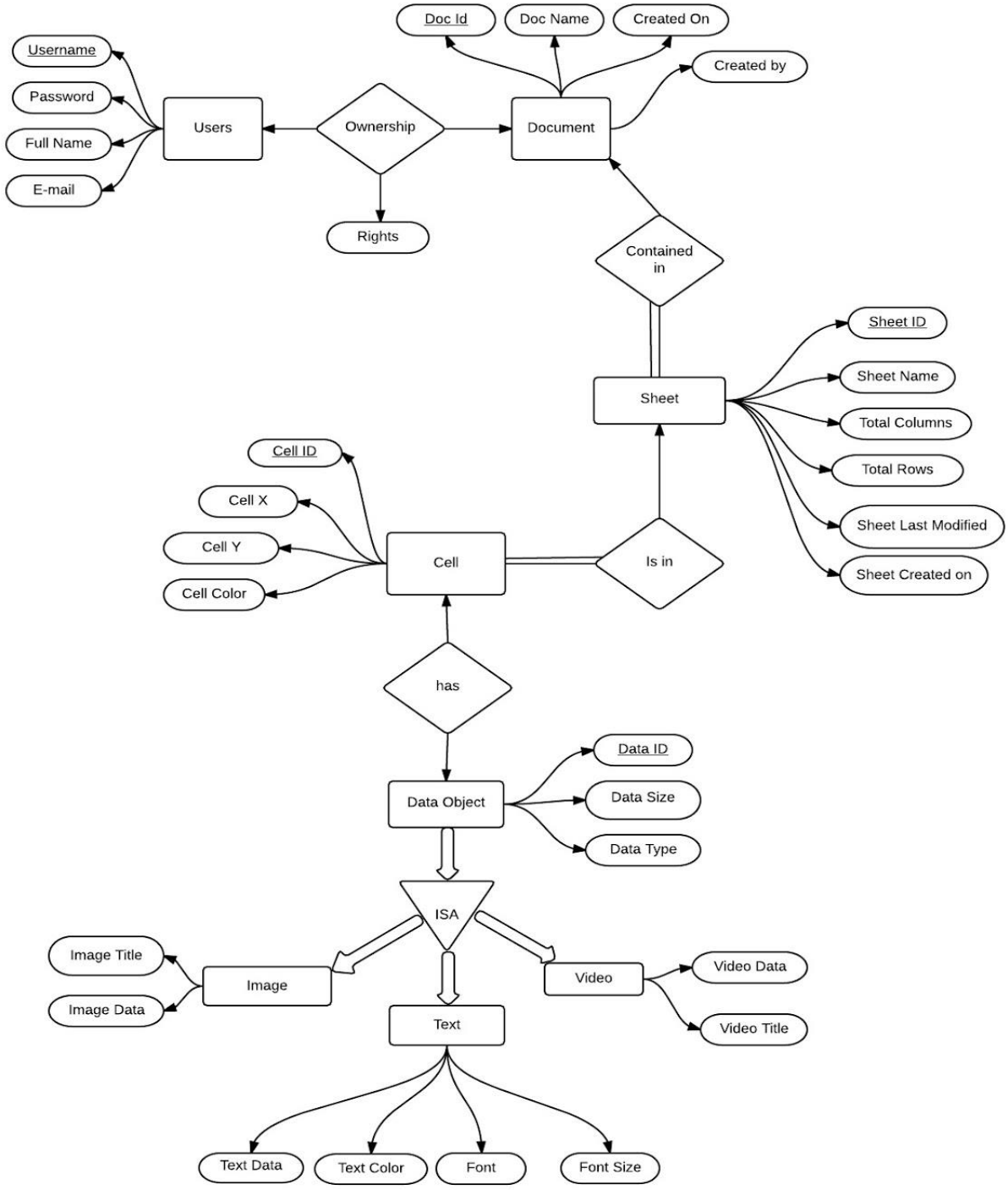
3) **Is in (Cell ID, Sheet ID)**

-binary relation which relates entities Sheets and Cell

4) **Has(Cell ID, Data ID)**

-binary relation which relates entities Cell and Data

C] ER Diagram:


Doc Id
Doc Name
Created On
Created by
Username
Password
Full Name
E-mail
Users
Ownership
Document
Rights
Contained in
Sheet ID
Sheet Name
Total Columns
Total Rows
Sheet Last Modified
Sheet Created on
Sheet
Cell ID
Cell X
Cell Y
Cell Color
Cell
Is in
has
Data ID
Data Size
Data Type
Data Object
ISA
Image Title
Image Data
Image
Video
Video Data
Video Title
Text
Text Data
Text Color
Font
Font Size

D] Functional Dependencies and Normalization:

As is obvious all super keys determine the tuple.

The ER model is sufficiently in normal form and complies with 3NF and BCNF rules.

We originally started with complex relations in 1NF which were further reduced into more relations using decomposition algorithms which preserve Functional Dependencies.

The Database Schema is in 3NF and is devoid of any redundancy in data representation.

Following is the detailed analysis of each relation along with their functional dependencies:-

---

Relation: **User (Username, Password, Full Name, Email)**

Username -> Password, Full Name, Email

Email -> Username, Password, Full Name

All attributes are in atomic form, and the relation is in Third Normal Form hence a “good” relation.

Relation: **Ownership (Username, Rights, Doc ID)**

Rights attribute is in atomic form, it can take values = {admin, read, edit}

Username, Doc ID -> Rights

All attributes are in atomic form, and the relation is in Third Normal Form hence a “good” relation.

Relation: **Document (Doc ID, Doc Name, Doc Created On, Doc Created By)**

Doc ID -> Doc Name, Created On, Created By

All attributes are in atomic form, and the relation is in Third Normal Form hence a “good” relation.

Relation: **Contained In (Doc ID, Sheet Id)**

The dependencies are trivial and hence it is in Third Normal Form, hence a “good” relation.

Relation: **Is_in (Sheet ID, Cell ID)**

The dependencies are trivial and hence it is in Third Normal Form, hence a “good” relation.

Relation: **Sheet (Sheet ID, Sheet Name, Total Columns, Total Rows, Sheet Last Modified, Sheet Created on)**

Sheet ID -> Sheet Name, Total Columns, Total Rows, Sheet Last Modified, Sheet Created on

---

Relation: **Cell (Cell ID, Cell X, Cell Y, Cell Color)**

Cell ID -> Cell X, Cell Y, Cell Color

Note: Cell X, Cell Y does not uniquely determine Cell Color because two cells with same location but in two different sheet can have different color.

Relation: **Has (Cell ID, Data ID)**

The dependencies are trivial and hence it is in Third Normal Form, hence a “good” relation.

Relation: **Data Object (Data ID, Data Size, Type)**

Type takes values = {Text, Image, Video}

Data ID -> Data Size, Type

Relation: **Image (Data ID, Image Data, Image Title)**

Data ID -> Image Data, Image Title

Relation: **Text (Data ID, Text Data, Text Color, Text Font, Font Size)**

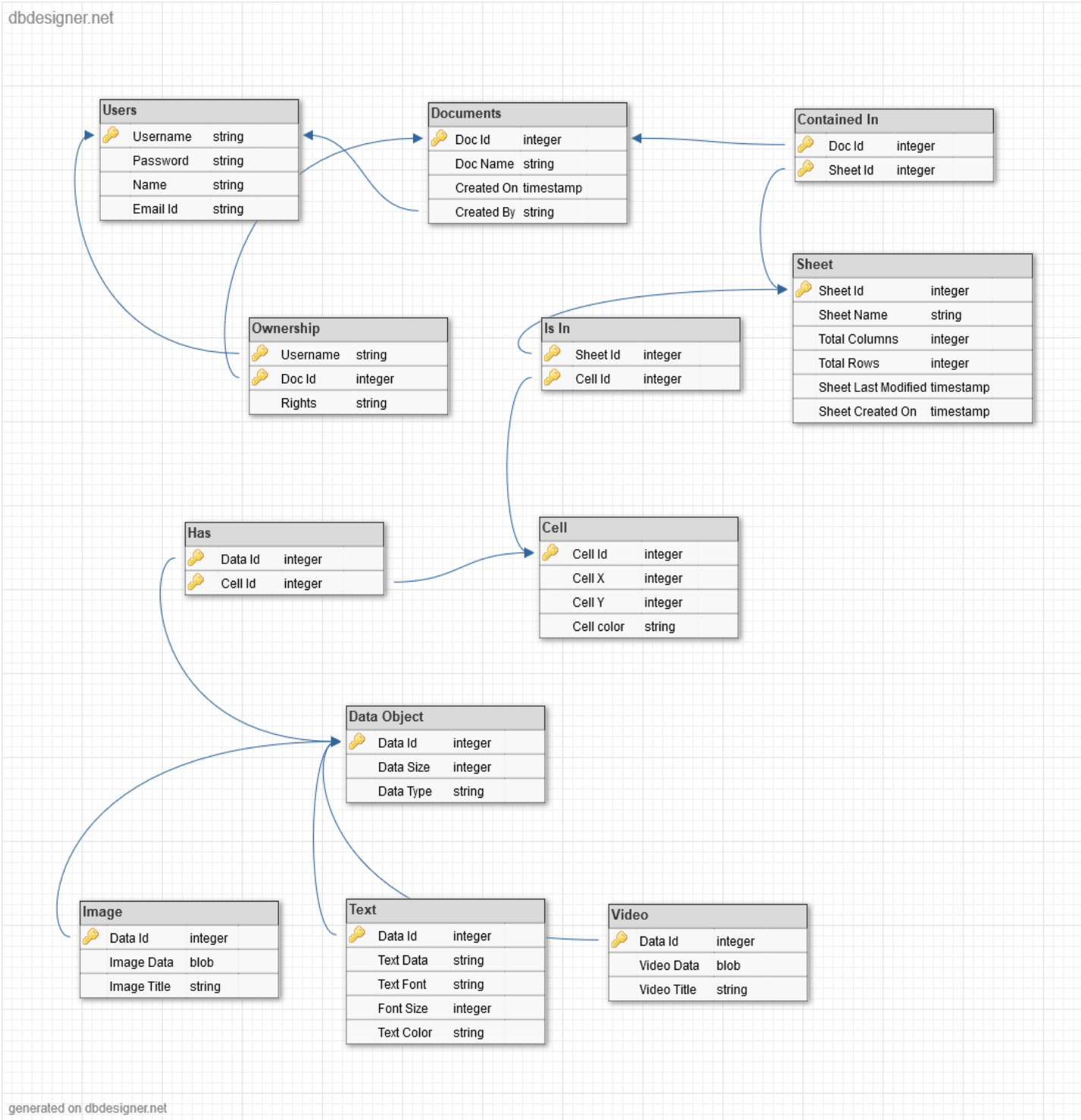
Data ID -> Text Data, Text Color, Text Font, Font Size

Relation: **Video (Data ID, Video Data, Video Title)**

Data ID -> Video Data, Video Title

All attributes are in atomic form, and the relation is in Third Normal Form hence a “good” relation.

E] Database Schema:



F] Assertions and Constraints:

- User Email Id should satisfy the standard regex form of an email address
- Creator of a document is made the Admin by default.
- Last modified time/date is greater than Created on time/date.
- Total Columns, Total Rows > 0 and Left Topmost Cell is (0,0) [Origin]
- Video Size <= 10MB, Image Size <= 2MB
- Password length >= 8, Username length <= 15
- Type of data is in {Text, Image, Video}
- Maximum length of text inside a cell is 1000 characters
- Rights can be of two type namely {Read, Edit}

G] Web Implementation Details:

Of course rendering documents to the user will require extensive interaction with the database as a result we will number of I/Os will be very high.

Django being an MVC (Model View Controller) framework, the relations in the database schema will be implemented as Models (same as classes in case of Java).

Each Relation in the Database Schema is modeled as a Class in Django Platform, each tuple in the relation is therefore an instance of the Class which we most familiarly know as a Class Object.

One of the reason of opting for Django Framework for this project was the availability of vast plethora of optimized inbuilt functions on such class objects which greatly facilitated coding and allowed us to focus more on the structural design aspect of the database.

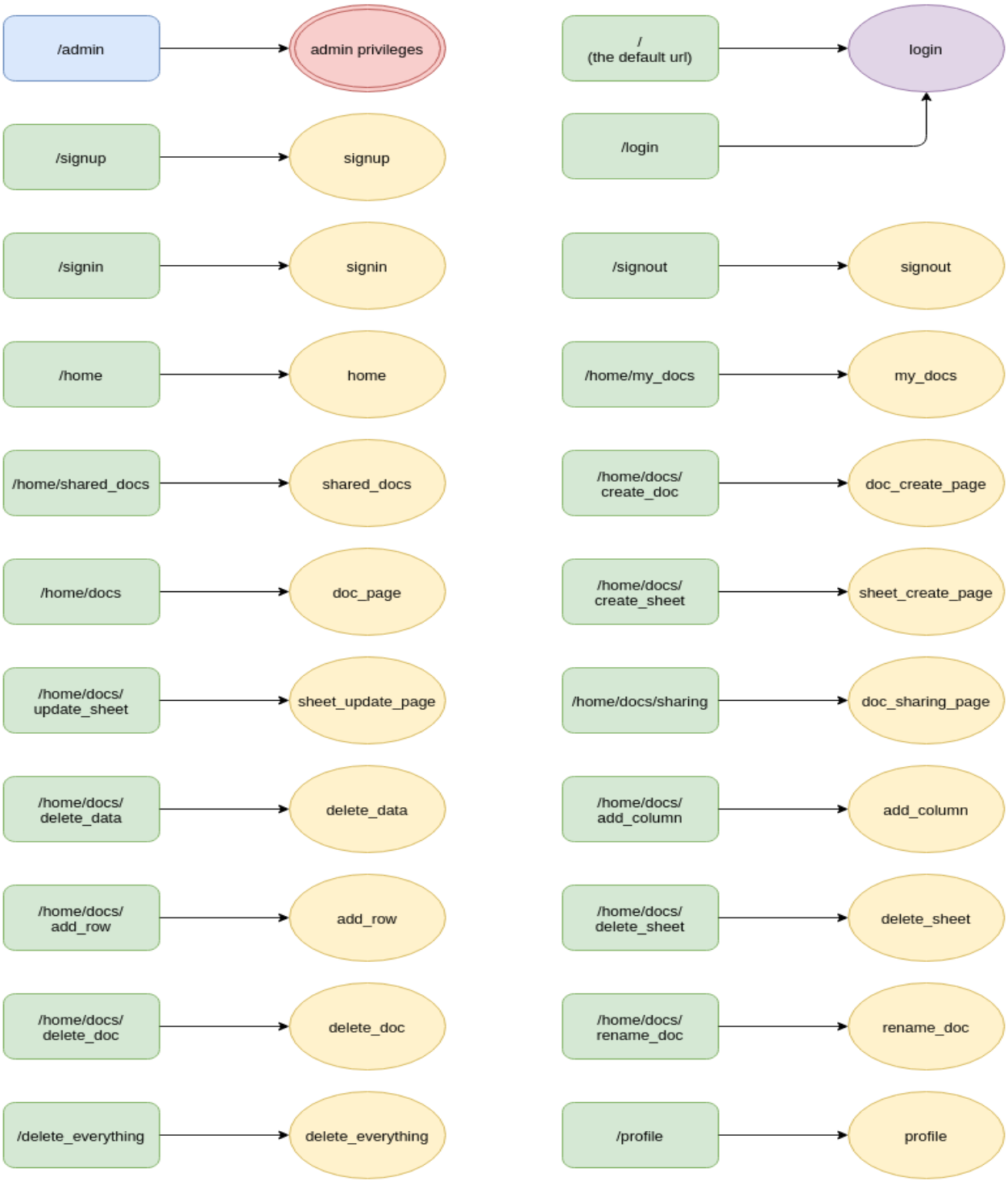
There are Three Major Files in the project which decide the control flow namely- url.py, view.py and model.py.

Url.py contains the url vs function mapping, View.py contains function definitions and Model.py contains the database definitions.

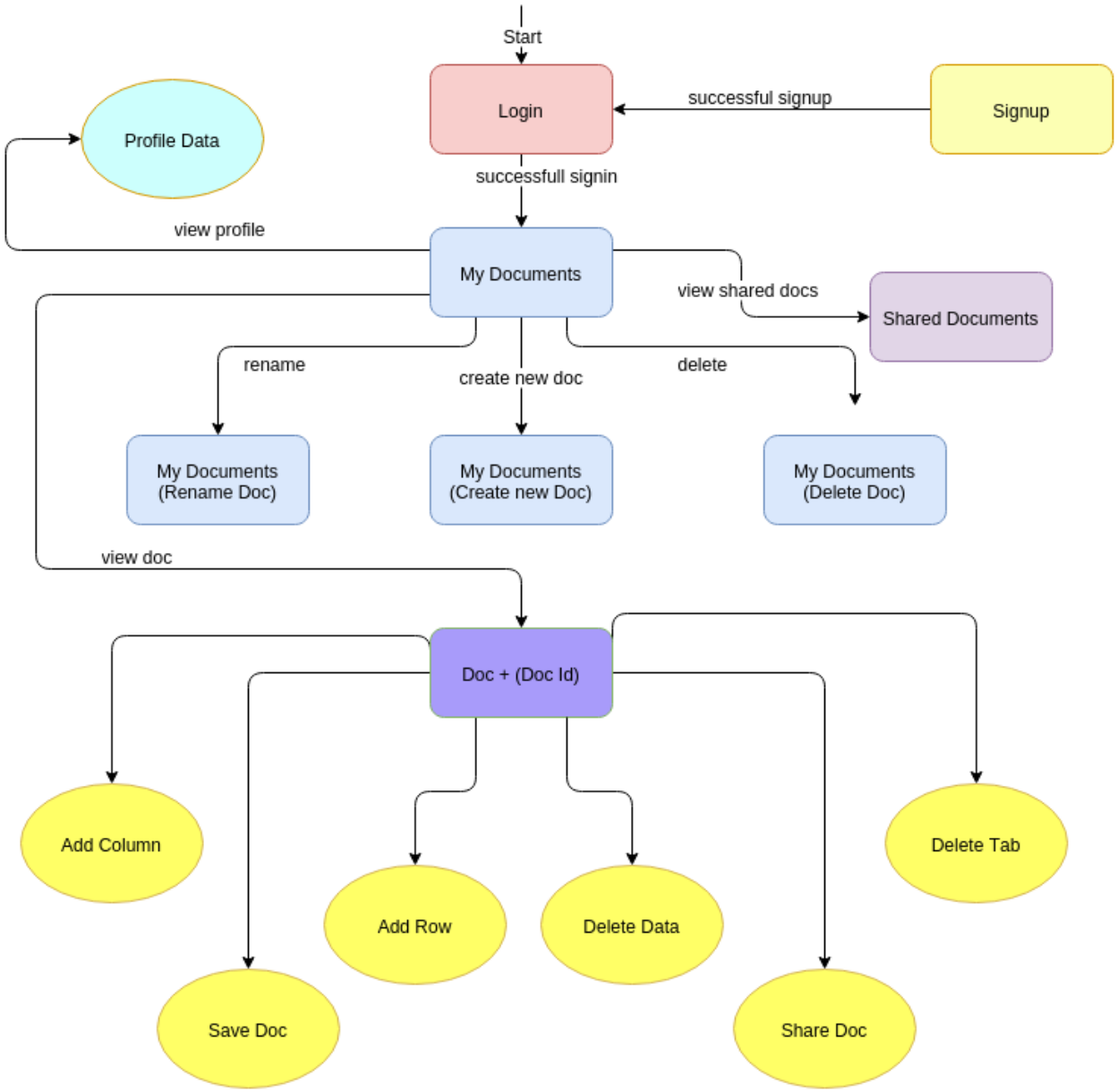
To give a brief overview of what we have created, we will start by explaining the urls in the project and the functions associated with each url.

The open source tool we have used is extensively documented here:https://www.djangoproject.com/

URL and Function mappings:



Sitemap:



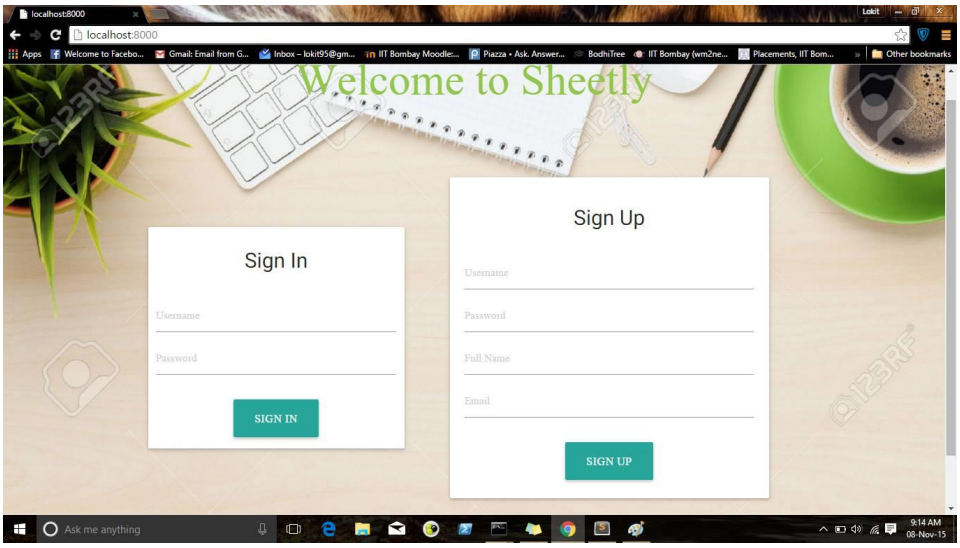
The functionality of each function can be easily guessed by its name, we will further list all the Database queries made by each function to further crisply explain the workings.

- **Login (request):** Redirects user to /home and renders login.html
- **Signin (request):** Authenticates the user credentials and gives user access accordingly
- **Signup (request):** Stores user credentials in the database,
    - Insertion in Users Relation
- **Signout (request):** Logs out the user and redirects to default page

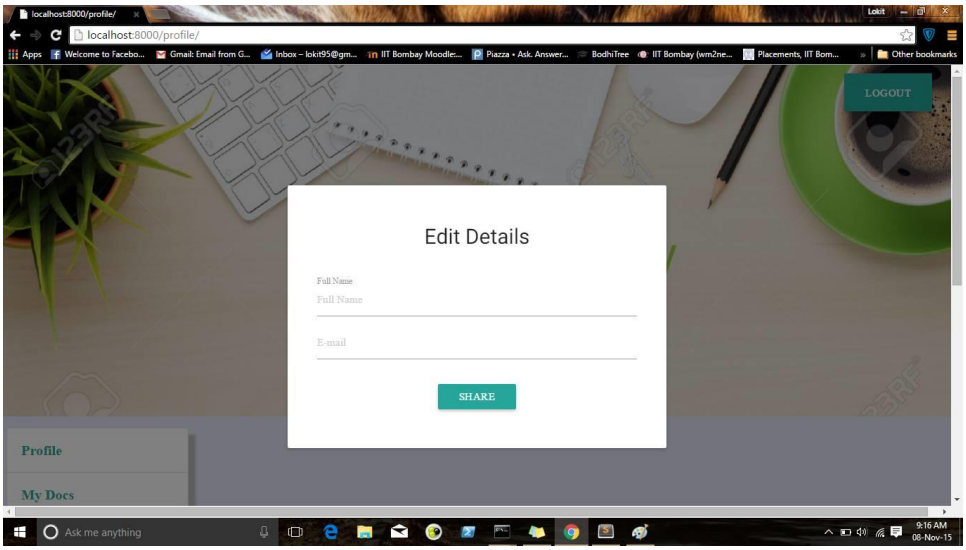
- **Home (request):** Redirects to /home/my_docs
- **My_docs (request):** Displays documents created by to the user
    - SELECT * FROM documents WHERE created_by=%s",[user.username])
- **Shared_docs(request):** Show the documents shared with the user
    - SELECT * FROM documents natural join ownership WHERE username=%s and created_by !=%s",[user.username,user.username])
- **Doc_create_page(request):** Creates Document for the user
    - Insertion in Documents Relation
- **Sheet_update_page(request, sheet_number):** Updates the sheet
    - Updation in Cell Relation
- **Doc_page(request, sheet_number):** Renders the Document webpage
    - SELECT * FROM sheets natural join contained_in WHERE doc_id = %s",[doc_id]
    - SELECT * FROM cell natural join is_in WHERE sheet_id = %s",[sheet.pk]
    - The above queries help in accessing sheets and correponding cells to be rendered.
- **Sheet_create_page(request,sheet_number):** Creates new sheet
    - Insertion in Cell Relation
    - Insertion in Is_in Relation
    - Insertion in Data_Object Relation
    - Insertion in Has Relation
    - Insertion in Text/Image/Video Relation
    - Insertion in Contained In Relation
    - Insertion in Sheets Relation
- **Doc_sharing_page(request,sheet_number):** Enables sharing document with other users
    - Updation/Insertion in Ownership Relation
- **Check_auth(request):** Security check on entered user credentials (used as a helping function by other functions)
- **Profile(request):** Checks Authorization using check_auth function and displays profile information.
- **Add_column(request,sheet_number):** Adds a new empty column to the sheet
    - Insertion in Cell Relation
    - Insertion in Is_In Relation
    - Updation in Sheets Relation

- **Add_row(request,sheet_number):** Adds a new empty row to the sheet
    - Insertion in Cell Relation
    - Insertion in Is_In Relation
    - Updation in Sheets Relation
- **Delete_data(request,sheet_number):** Deletes the target cell along with shifting of remaining cells accordingly in the sheet.
    - Deletion in Cell Relation
    - Updation in Cell Relation
    - Helping query used: SELECT * FROM cell natural join is_in WHERE sheet_id = %s",[sheet.pk].
- **Delete_sheet(request,doc_number):** Deletes the Sheet
    - SELECT * FROM cell natural join is_in WHERE sheet_id = %s",[sheet.pk]
    - Deletes all the cells and in the end the sheet using above query
- **Delete_doc(request,doc_number):** Deletes the Document
    - SELECT * FROM sheets natural join contained_in WHERE doc_id = %s",[doc_id]
    - SELECT * FROM cell natural join is_in WHERE sheet_id = %s",[sheet.pk]
    - Deletes cells, sheets and in the end the document using above queries.
- **Rename_doc(request,doc_number):** Renames the Document
    - Updation in Documents Relation
- **Delete_everything(request):** Destroys all Relations
    - Deletes Cell Relation
    - Deletes Sheets Relation
    - Deletes Documents Relation

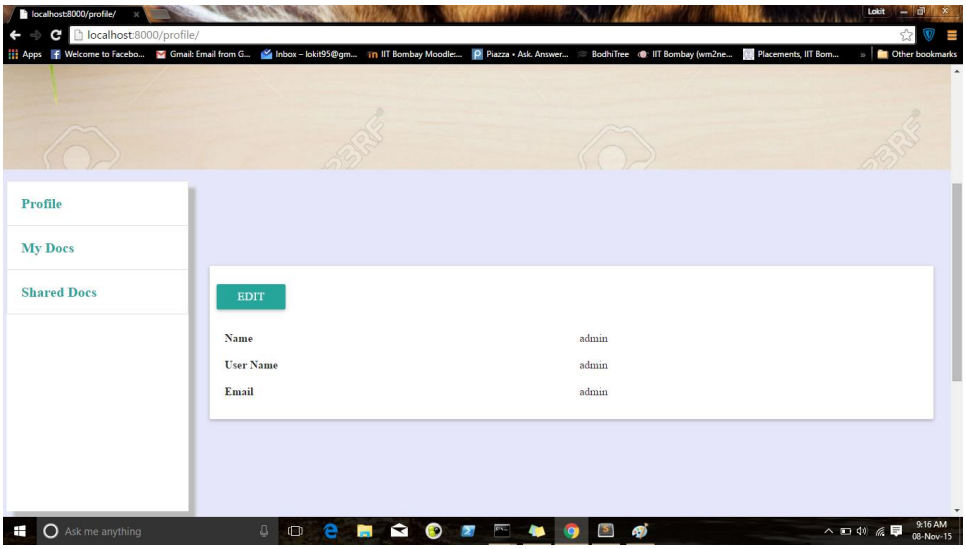
H] Screenshots:



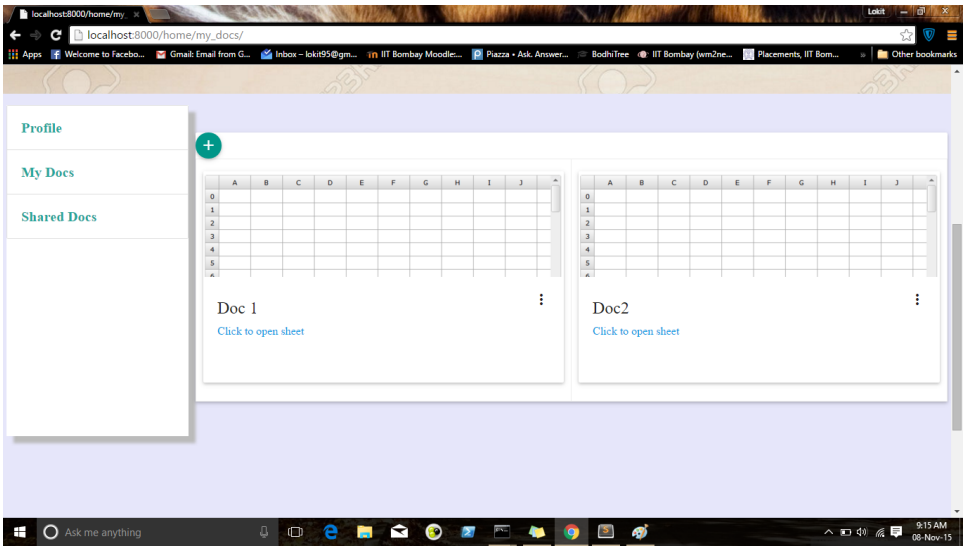
Login



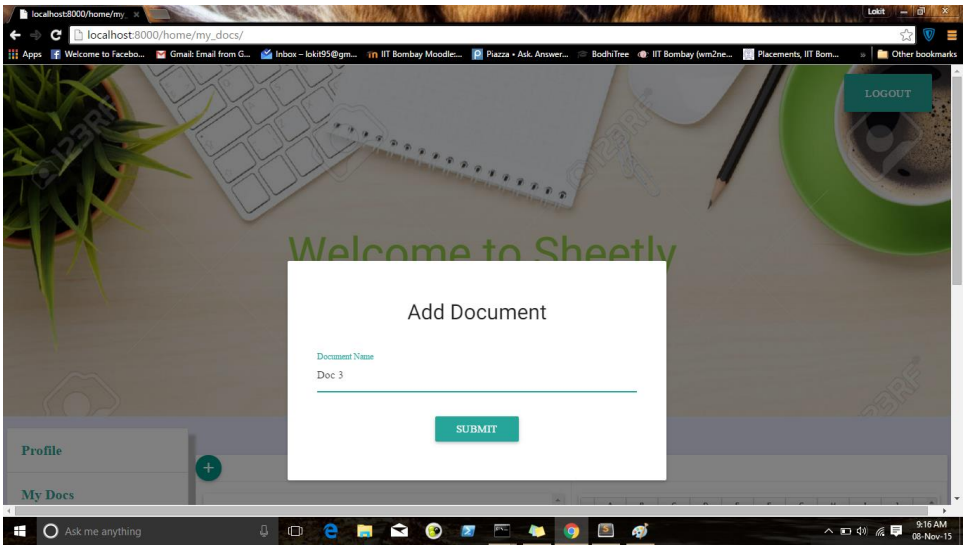
Edit Profile



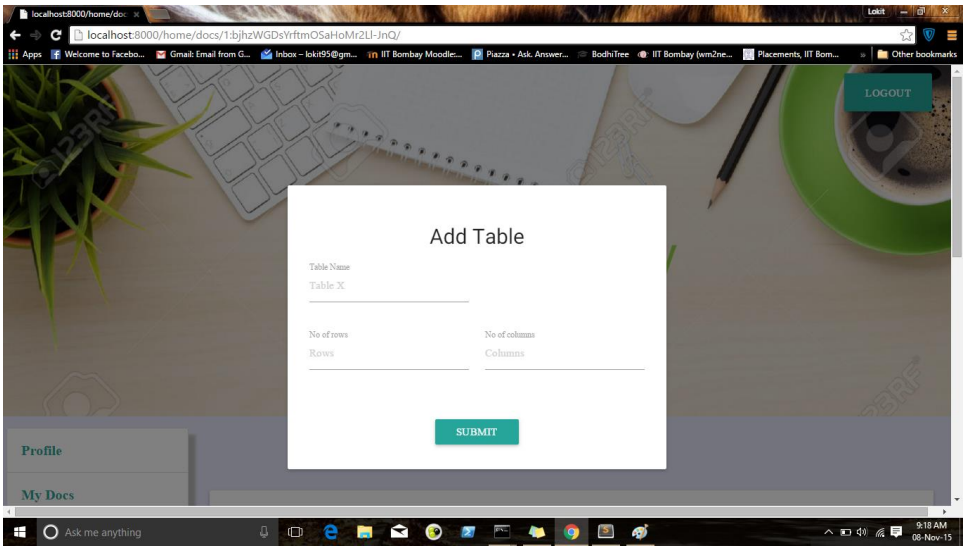
View Profile



Home Page

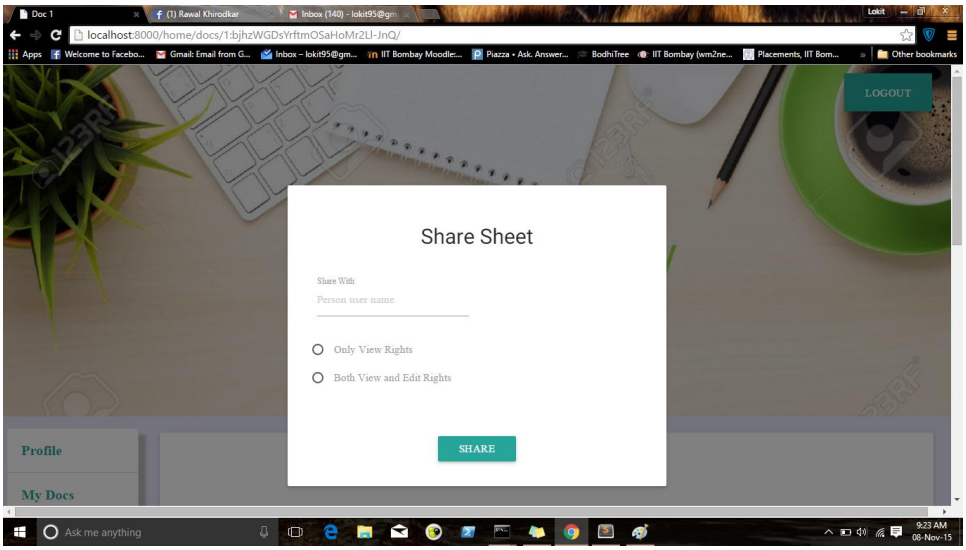


Add Doc

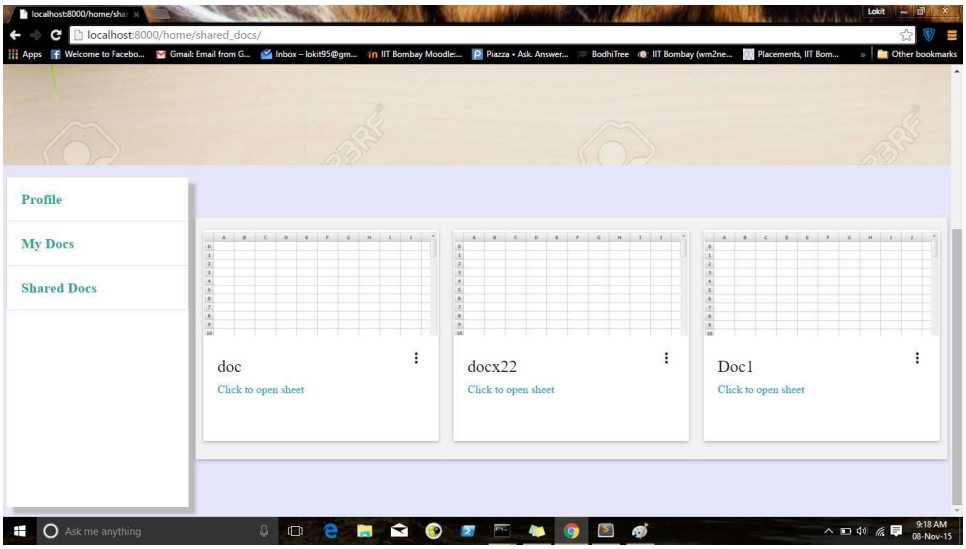


Add Sheet

Doc Details



Share Sheets



Shared Docs

I] Conclusion / Further Improvements:

Further optimization would be to decrease the lag between generation of new Database Relations, also one can implement a feature of real time collaborative editing of documents.

The database is asynchronous as we are not doing real time processing, using Ajax would enable to create a more robust version of this project.