



**REII 413**

**Practical**

by:

S. Jacholke 23671750

H.K. Blackie 23377852

submitted in pursuit of the degree

**BACHELOR OF ENGINEERING**  
in  
**COMPUTER AND ELECTRONIC ENGINEERING**

**North-West University Potchefstroom Campus**

Lecturer: Mr. A.J. Alberts  
Potchefstroom  
June 2, 2015  
version 1.0

It all starts here



## CONTENTS

---

<b>i</b>	<b>INTRODUCTION</b>	<b>1</b>
1	INTRODUCTION	2
2	PROBLEM STATEMENT AND AIM	3
3	METHODOLOGY	4
<b>ii</b>	<b>LITERATURE</b>	<b>5</b>
4	LANGUAGES	6
4.1	HTML . . . . .	6
4.2	CSS . . . . .	6
4.3	JavaScript . . . . .	7
4.4	PHP . . . . .	7
4.5	SQL . . . . .	7
5	DATABASE	9
5.1	Purpose of the database . . . . .	9
5.2	Single- or multi-user database . . . . .	10
5.3	Types of information . . . . .	10
5.4	Database Tables . . . . .	10
5.5	Primary Keys . . . . .	10
5.6	Table relationships . . . . .	11
5.7	Normalization . . . . .	11
<b>iii</b>	<b>DESIGN</b>	<b>12</b>
6	DATABASE DESIGN	13
6.1	Logical Files . . . . .	14
6.2	Folders . . . . .	16
6.3	Virtual Files . . . . .	16
6.4	Permissions . . . . .	17
6.5	Users . . . . .	17
6.6	User_plans . . . . .	17
6.7	Public_folders . . . . .	18
6.8	Logins . . . . .	18
6.9	File_uploads . . . . .	18
6.10	File_downloads . . . . .	18
6.11	Normalization . . . . .	19
7	WEBSITE DESIGN	21
7.1	User management . . . . .	21
7.2	File management . . . . .	21
7.3	Profile . . . . .	23
8	CONCLUSION	26

## LIST OF FIGURES

---

Figure 1	Crow's Foot ERD Notation . . . . .	11
Figure 2	Entity relationship diagram of database . . . . .	15
Figure 3	First attempt at designing the virtual files table . . . . .	20
Figure 4	Registration page . . . . .	21
Figure 5	Home page . . . . .	22
Figure 6	Browsing page listing folders . . . . .	22
Figure 7	Moving of files . . . . .	23
Figure 8	Uploading of files . . . . .	24
Figure 9	Navigation icons . . . . .	24
Figure 10	Public folder sharing . . . . .	25
Figure 11	Profile page . . . . .	25

## LIST OF TABLES

---

## LISTINGS

---

## ACRONYMS

---

**KISS** Keep it simple, stupid

**AJAX** Asynchronous JavaScript and XML

**HTML** HyperText Markup Language

**PHP** PHP Hypertext Preprocessor

**SQL** Structured Query Language

**CSS** Cascading Style Sheets

**DBMS** Database Management System

**XHTML** Extensible Hypertext Markup Language

**DHTML** Dynamic Hypertext Markup Language

**JS** JavaScript

## Part I

### INTRODUCTION

## INTRODUCTION

---

The 21st century is time where the Internet and cloud computing took the world by storm. People feel they need to be connected to the outside world through the Internet the whole day. Smartphones enable users to access things such as Internet, email and documents quicker and more effective without any trouble. Because the world is moving towards mobile and cloud computing, a need exist for people to access their files on their computer from anywhere. Using websites such as Google Drive, Dropbox, OneDrive and iCloud Drive, people are able to upload and store their files to access from any device without the need for a computer. These websites restrict users to a certain amount of free storage, after which they need to buy more space to store files.

To store the files using the website and a server or a host, certain structures need to be put in place. This includes the server and the client side of the website. The website needs to store user information and profiles, files such as documents, music, images and videos. These website enable the user to share and collaborate with other friends on the website. The website also handles sessions and file permissions. Behind the scenes, there is a big database running on a server which handles and stores all the files. The database is managed through some sort of database manager and updated regularly.

Programming languages such as HyperText Markup Language ([HTML](#)), JavaScript, PHP Hypertext Preprocessor ([PHP](#)) and MySQL enables the designer of host to create websites to store these files. These languages are used so that the user may interact with the server, and using [PHP](#) and MySQL to communicate with the sever to send and receive certain information. For the purpose of this practical, the student must develop a file storage and saving website, using the knowledge obtained in REII 413, and other programming languages to develop this website. The process for developing this website will be discussed and collaborated on.

## PROBLEM STATEMENT AND AIM

---

The purpose of this practical is to develop a file storage website. Users must be able to create profiles and upload files to a server for safe keeping. [PHP](#) and MySQL must be used to do this. The website must be user friendly and designed using [HTML](#) and Cascading Style Sheets ([CSS](#)) for styling.



## METHODOLOGY

---

The website must have the following features:

- User must be able to register and log into the page.
- The user must be able to create directories.
- The user must be able to upload files to the website. The developer must make use of Asynchronous JavaScript and XML ([AJAX](#)) to do this.
- The website must be able to move or delete files and directories.
- Must be able to administer user groups.
- Files must have the ability to be shared privately or publicly viewable.
- [CSS](#), JavaScript and other web based languages may be used to style the website.
- Limits must be placed on the amount of online storage available.

Using these specifications as a guide, the website can be designed accordingly. All these data must be kept in a database and managed through some database manager.

Part II

LITERATURE

## LANGUAGES

---

Before continuing with the development of the website, there must first be understood why certain programming and scripting languages are used and what their purpose is. The following scripting languages will be looked at:

- [HTML](#)
- [CSS](#)
- JavaScript ([JS](#))
- [PHP](#)
- Structured Query Language ([SQL](#))

These are the five basic languages used in modern website development.

### 4.1 HTML

[HTML](#) is used to develop a website. Web browsers can read [HTML](#) files and render them into visible or audible web pages.

Browsers do not display the [HTML](#) tags and scripts, but use them to interpret the content of the page. [HTML](#) describes the structure of a website semantically along with cues for presentation, making it a mark-up language, rather than a programming language.

[HTML](#) elements form the building blocks of all websites. [HTML](#) allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts written in languages such as JavaScript which affect the behaviour of [HTML](#) web pages.

With the development of HTML5, enabled developers to upload bigger files for web storage. It also enabled document editing, drag-and-drop, cross-document messaging and other APIs.

### 4.2 CSS

[CSS](#) is a style sheet language which is used to describe the look and formatting of any document written in a markup language such as [HTML](#), Extensible Hypertext Markup Language ([XHTML](#)), etc.

[CSS](#) is designed primarily to enable the separation of document content from document presentation, including elements such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics.

The formatting is done by specifying a relevant [CSS](#) in a separate .css file. For each matching [HTML](#) element, it provides a list of formatting instructions. A [CSS](#) rule might for example specify that all heading 1 elements should be bold, leaving pure semantic [HTML](#) markup that asserts 'this text is a level 1 heading' without formatting code such as a <bold> tag indicating how such text should be displayed.

#### 4.3 JAVASCRIPT

[JS](#) is a dynamic programming language. It is used as part of web browsers which allows client-side scripts to interact with the user. [JS](#) is used to control the browser, communicate with the user, alter displayed document content and server-side network programming.

[JS](#) is most commonly used to add client-side behaviour of Dynamic Hypertext Markup Language ([DHTML](#)) pages. Scripts are embedded within the [HTML](#) page or it includes to interact with the

Document Object Model (DOM) of the pages. Some uses of [JS](#) may include:

- Loading new page content.
- Submitting data to a server by means of [AJAX](#) without page reloading.
- Animation of page elements (fading, resizing, moving, etc.)
- Interactive content such as games, audio, video.
- Validating web form values before submitting to a server.
- Handles data such as cookies, ad tracking and other personalization purposes

#### 4.4 PHP

[PHP](#) is a server-side scripting language for web development. [PHP](#) generally runs on a web server. A request file is sent to a server, if it contains any [PHP](#) code, it gets executed, and an appropriate output file is sent back. [PHP](#) was originally designed to create dynamic web pages, however, the focus moved to server-side scripting. [PHP](#) provides dynamic content from a web server to a client.

#### 4.5 SQL

[SQL](#) is a programming language which is designed for managing data in a Database Management System ([DBMS](#)), of which MySQL forms part of. [SQL](#) consists of a data definition language and a data manipulation language.

[SQL](#) includes data inset, query, update and delete, schema creation and modification and data access control.

[SQL](#) enables a database designer to interact with the database and to perform complex operations on the database. Combined with [PHP](#), [SQL](#) can be used to communicate to a server through a web page to send and receive

data from the server. It help to exchange data between the client and the web-server.

## DATABASE

---

Developing a web-site for file management and storage, a robust database must be created to handle user registration, passwords, permissions, files and other relevant information.

To design a database a few guidelines needs to be followed [x] to help in the design process. Before developing the database, the following aspects need to be considered [1]:

- Determine the purpose of the database.
- Single- or multi-user database.
- Determine all the types of information needed to be stored.
- Divide important information into major tables.
- Determine which information needs to be stored in which table as columns.
- Specify primary keys.
- Specify table relationships.
- Analyse the design for errors, and finally
- Normalization of the database to remove database ambiguities.

### 5.1 PURPOSE OF THE DATABASE

The purpose of the website is to manage and store user files for easy access from any devices which has an internet connection. Therefor, the database must be capable of storing user information and registration details, handling files and link the uploaded files to certain users.

The user must also be able to share his/her files between other users to give them public or private access to the files and folders. Therefor the purpose of the database is to:

- Store user information.
- Store folder and file names.
- Link folders and files to users.
- Manage folder and file permissions.
- Manage file uploads and downloads.
- Link user data usage plans to certain users which requires more storage.

## 5.2 SINGLE- OR MULTI-USER DATABASE

Another thing the consider is to decide whether the database will be single-user or multi-user database.

A single-user database allows only one user at a time to access the database. Other users must wait for their turn before attempting to access the data.

A multi-user database support multiple users at the same time.

It is important for a website's database to be designed for multi-user purposes. Users would like the log into their profiles to upload or download files, and no one would like to wait for their turn.

## 5.3 TYPES OF INFORMATION

From a user registration point, it is important to know who the user is, therefore gather information such as his/her name, surname, email, and password is information which can be grouped later on in a table such as user\_info

## 5.4 DATABASE TABLES

In the [SQL](#) environment, a schema is a group of database object, such as tables and indexes that are related to each other. Usually, the schema belongs to a single user or application (website session).

A single database can hold multiple schemas which belongs to different users. On each user profile creation, a schema is create which is an instance of that database [1].

Schemas are useful in that they group tables by owner and therefor enforces a first level of security by allowing the user to see only the tables or data belonging to them.

Using the required information, there can be decided which types of information can be grouped together to form appropriate tables.

## 5.5 PRIMARY KEYS

Each table must have a primary key. A primary key is an attribute that uniquely identifies any given row in a table [2]. In a table USER\_INFO, a good primary key would be the user's username.

Usually a username is unique and identifies all the details about that person. Using a USER\_ID\_NO can also help to uniquely identify a row in a table. The primary key of a table is usually the foreign key of the next table.

The foreign and primary keys are usually used to link tables to each other to establish certain relationships between the tables in a database. It is also important to avoid ambiguity within a database. It may happen that too much information is duplicated within a database, but this can be sorted out by normalizing tables.





Symbol	Meaning
	One—Mandatory
	Many—Mandatory
	One—Optional
	Many—Optional

Figure 1: Crow's Foot ERD Notation

## 5.6 TABLE RELATIONSHIPS

Relationships within a database can be described as being a one-to-one (1:1), one-to-many (1:\*), many-to-one (\*:1) or many-to-many (\*:\*). Using these four possible combinations, the relationship between tables can be created. For example, one user may own many files, but one file is owned by only one user (if the files are not shared by multiple users).

Or one user may only have one storage plan, and a storage plan type may be owned by many users.

The Crows-foot model is a relationship model which is widely used in ER (entity-relationship) modelling of databases. Figure 1 shows the Crows-foot relational model.

## 5.7 NORMALIZATION

Database normalization is the process of organizing the attributes and tables of a relational database. This is done to minimize the data redundancy. It involves the decomposition of a table into smaller less redundant tables without losing any information.

Normalizations helps with data isolation. When data needs to be added to a table, deleted or modified, it can just be done in one table and then propagate through the rest of the database. It helps to avoid update, insertion, and deletion anomalies[3].



Part III

DESIGN

## DATABASE DESIGN

---

The steps mentioned in section 3 was used to help design the database for the website. The purpose of the database is stated in section 3.1.

It was decided to design a multi-user database. When users are using the website, it would be preferable to upload and fetch files from the database at the same time without the need to wait for another user to finish their session.

The following types of information will be needed to store within the database:

- User first name
- Last name
- Email address
- Password
- Folder names
- File names
- Folder and file paths
- File types and sizes
- Folder and file permissions
- Uploads and downloads register
- Original file and folder owners (sharing)

This information can be stored in appropriate tables within the database. The follow tables can be derived from the information above:

- USERS
- USER\_PLANS (Limit storage space)
- FOLDERS
- PUBLIC\_FOLDERS
- FILES
- PERMISSIONS
- FILE\_UPLOADS
- FILE\_DOWNLOADS
- LOGIN\_INFORMATION (Website visit count per user)

Within each of these tables, the following primary keys are chosen to uniquely identify an attribute within the table:

- USERS (User information)
  - USER\_ID

- USER\_PLANS (Data storage plan (drive size))
  - PLAN\_ID
- FOLDERS (Folders to store files)
  - FOLDER\_ID
- PUBLIC\_FOLDERS (Share folders)
  - FOLDER\_ID (The same id as in the table folders)
- FILES (Uploaded file information)
  - FOLDER\_ID
- PERMISSIONS (Files read, write and ownership identification)
  - FOLDER\_ID
  - USER\_ID
- FILE\_UPLOAD (Which file has been uploaded, setting read, write, ownership info. determining file size, etc.)
  - ID
- FILE\_DOWNLOAD
  - ID

For the practical it was decided that a user is the owner of a folder, and that the folder is the unit for sharing. As such folders can either be publicly shared (to all users and even non-users of the site) or given permissions to specific users of the site.

Figure 2 shows the entity relationship model of the database. This is the backend used by the website.

In the following sections the basic design and functionality of each table will be discussed.

## 6.1 LOGICAL FILES

This table lists the data for the files actually stored on the hard disk. It was decided to only store a single file if multiple users upload the same file.

This is incorporated by hashing each file. A hash function maps data of arbitrary size to data of fixed size. A good hash has no collisions or as little as possible. An ideal cryptographic hash is a function that is considered practically impossible to invert. Practically a cryptographic function may hash two different input to the same output, resulting in a collision, although the probability of this is extremely small.

The cryptographic hash SHA1 was used to hash each file's contents to a *unique* value. This hash value is used to check whether a file already exists on the hard drive, if the file already exists it is not stored again on the hard disk, and instead only a reference to it is made.

The table logical files is described by the following fields:

- file\_id - the primary key

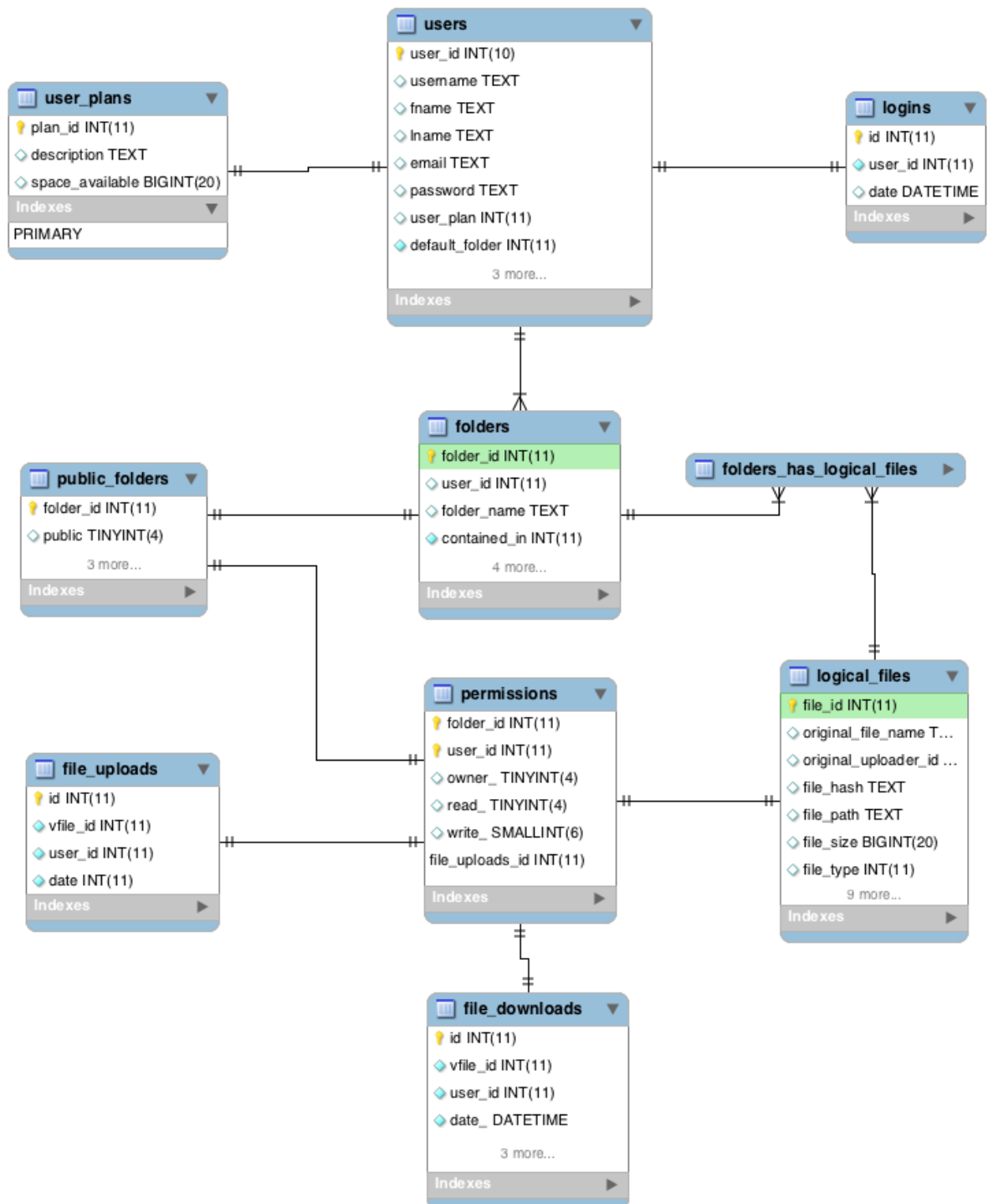


Figure 2: Entity relationship diagram of database

- `original_file_name` - The name of the file when it was uploaded
- `original_uploader_id` - The id of the user whom uploaded the file. Foreign key to users
- `file_hash` - The SHA1 hash of the file, as discussed above
- `file_path` - The path where the file is stored on the hard disk
- `file_size` - The size of the file on the hard disk
- `last_modified` - The date the file was uploaded.

In order to check whether a file already exists on the hard disk both the hash and the file size is used in order to reduce the amount of collisions. For a large scale site a longer hash function with a smaller collision rate should be used.

## 6.2 FOLDERS

This table lists the folders of the site. A folder is the basic entity which can be shared to other users and is owned by a single user.

The table is described by the following fields:

- `folder_id` - Primary key
- `user_id` - Owner of the folder. Foreign key to users
- `folder_name` - The name of the folder
- `contained_in` - The name of the parent directory in which the folder resides. Foreign key to Folders

As can be seen, the field `contained_in` is a foreign key to the same table, resulting in a circular reference. A folder that is not contained by any folder is known as the user's home folder and has the `contained_in` field set as -1.

The choice for a circular reference was made as it enables the folders to be easily moved and renamed. A circular reference is commonly a poor design construct however in this case it simplifies the database.

The location of folders is not stored on the hard disk and instead in the database as it enables file and folder manipulation. Should a site require distributed file sharing only a single table then needs to be modified.

## 6.3 VIRTUAL FILES

Virtual Files describes the relationship between a file on the hard disk, the name of the file and the location of the file in the user's path.

The table is described by the following fields:

- `vfile_id` - Primary key
- `pfile_id` - Foreign key to logical files
- `folder_id` - Foreign key to folders
- `file_name` - the name of the file for the user

## 6.4 PERMISSIONS

The table `permissions` describe the user sharing of folders. It stores the information which indicate which users have read and write access to certain folders.

The table is described by the following fields:

- `folder_id` - Foreign key to folders
- `user_id` - Foreign key to users
- `owner_` - Indicate whether the user is the owner of the folder
- `read_` - Indicates whether the user has read access to the folder
- `write_` - Indicates whether the user has write access to the folder

The primary key of this table is a composition of `folder_id` and `user_id`. The table lists the permissions for a folder for each user.

Only users with read access may download the contents of a folder. Only users with write access may upload, rename, move or delete the contents of a folder.

The choice was made to not allow access to sub folders by default even though the parent folder is shared. Each sub-folder needs to be explicitly shared.

## 6.5 USERS

This table stores the users of the database.

The table is described by:

- `user_id` - Primary key
- `username` - The alias or handle of the user
- `fname` - The first name of the user
- `lname` - The last name of the user
- `email` - The email address of the user
- `password` - A cryptographic hash of the password
- `user_play` - Foreign key to user plans
- `default_folder` - The home directory of the user. Foreign key to folders

## 6.6 USER\_PLANS

This table stores the storage quotas of different membership plans of the site. The storage quota restricts the amount of files a user may upload

The table is described by:

- `plan_id` - Primary key
- `description` - Short text describing the plan
- `space_available` - The maximal amount of storage space the user may use

This table restricts the amount of data a user may have on their account. Note that this describes the total sizes of all virtual files the user has uploaded. The actual amount of hard disk space used by the user would be strictly equal or less than that amount.

## 6.7 PUBLIC\_FOLDERS

This table lists which folders may be accessed by all users and non-users of the site. The term public in the context of the site represents such type of folder access.

The table is described by the following fields:

- folder\_id - Primary key
- public - Indicates if everyone has access to the folder

## 6.8 LOGINS

This table list the date and time when a user logs in

The table is described by the following fields:

- id - Primary key
- user\_id - Foreign key to users
- date - The date and time a user logged in

## 6.9 FILE\_UPLOADS

This table list the date and time when a file was uploaded

The table is described by the following fields:

- id - Primary key
- vfile\_id - Foreign key to logical files
- user\_id - Foreign key to users
- date - The date and time a user logged in

## 6.10 FILE\_DOWNLOADS

This table list the date and time when a file was downloaded

The table is described by the following fields:

- id - Primary key
- vfile\_id - Foreign key to logical files
- user\_id - Foreign key to users
- date - The date and time a user logged in

## 6.11 NORMALIZATION

Third normal form is the most common form of normalization in a database. It reduces the duplication of data and ensures referential integrity by ensuring:

- The entity is in second normal form
- The attributes are determined only by the candidate keys of the table and not by non-prime attributes

A table is in second normal form when:

- The entity is in first normal form
- Every non-prime attribute of the table is dependent on the whole of every candidate key

A table is in first normal form when

- The entity contains only atomic values
- There are no repeating groups.

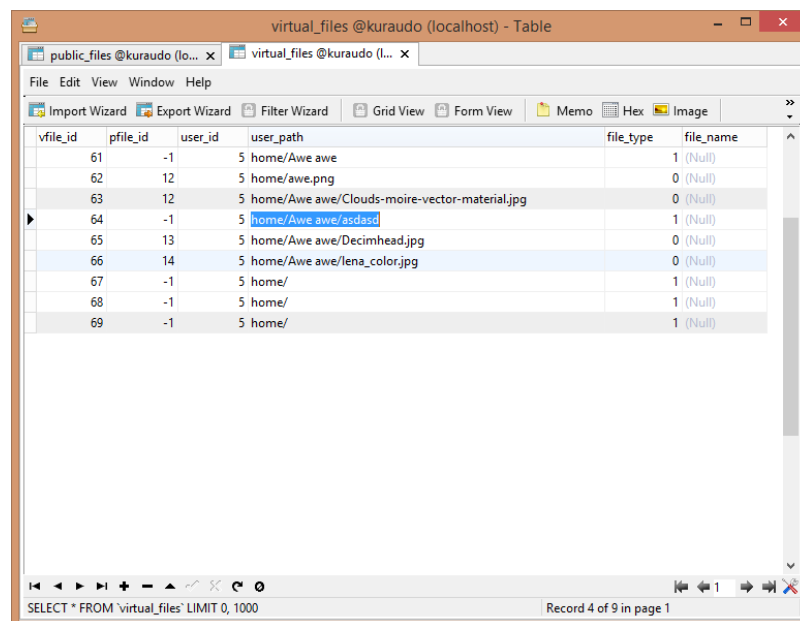
The tables were designed from the start in third normal form by satisfying the above properties.

A first design attempt resulted in the virtual files containing a complex `user_path` field that stores the entire path of a folder for example: *home/files/new/image.png*. This does not satisfy the requirement of first normal form where an entity should only contain atomic values. This is shown in figure 3.

The choice was made to include an additional table for Folders. Even though it contains a circular reference it still satisfied the requirements of third normal form.

Using the normalized tables allow easier and less redundant use of data from the database.





vfile_id	pfile_id	user_id	user_path	file_type	file_name
61	-1	5	home/Awe awe	1 (Null)	
62	12	5	home/awe.png	0 (Null)	
63	12	5	home/Awe awe/Clouds-moire-vector-material.jpg	0 (Null)	
64	-1	5	home/Awe awe/asdasd	1 (Null)	
65	13	5	home/Awe awe/Decimhead.jpg	0 (Null)	
66	14	5	home/Awe awe/lena_color.jpg	0 (Null)	
67	-1	5	home/	1 (Null)	
68	-1	5	home/	1 (Null)	
69	-1	5	home/	1 (Null)	

SELECT \* FROM 'virtual\_files' LIMIT 0, 1000

Record 4 of 9 in page 1

Figure 3: First attempt at designing the virtual files table

## WEBSITE DESIGN

---

The website is a simple file sharing site where users may upload, download, move, rename and delete files. In addition files may be publicly shared to everyone or only to certain users. Each user has a quota on the total size of all files uploaded.

The website used the database backend as discussed in section 6.

The website utilizes Twitter's Bootstrap framework. Bootstrap is a popular [HTML](#), [CSS](#) and JavaScript framework that allows for responsive web application. By utilizing the Bootstrap framework less time needs to be spent on tweaking [HTML](#) and [CSS](#) components and allows the site to be viewed pleasantly on a wide variety of devices.

In order for the site to be responsive the calls for file manipulation is sent by JavaScript post calls. By using [AJAX](#) the site does not need to be refreshed after each action as data can be sent and received asynchronously.

### 7.1 USER MANAGEMENT

Users can be created by using the registration page as is shown in figure 4.

When a user is created a they may login (Figure 5). Once a user has logged in they will be transferred to the browse page.

### 7.2 FILE MANAGEMENT

In the browse page users may upload, move, rename and delete files and folders.

Kuraudo Please sign in

Please sign up

First Name Last Name

Username

Email Address

Password Confirm Password

☐ I agree By clicking [Register](#) you agree to the [Terms and Conditions](#) set out by this site, including our [Cookie Use](#).

REGISTER

Figure 4: Registration page

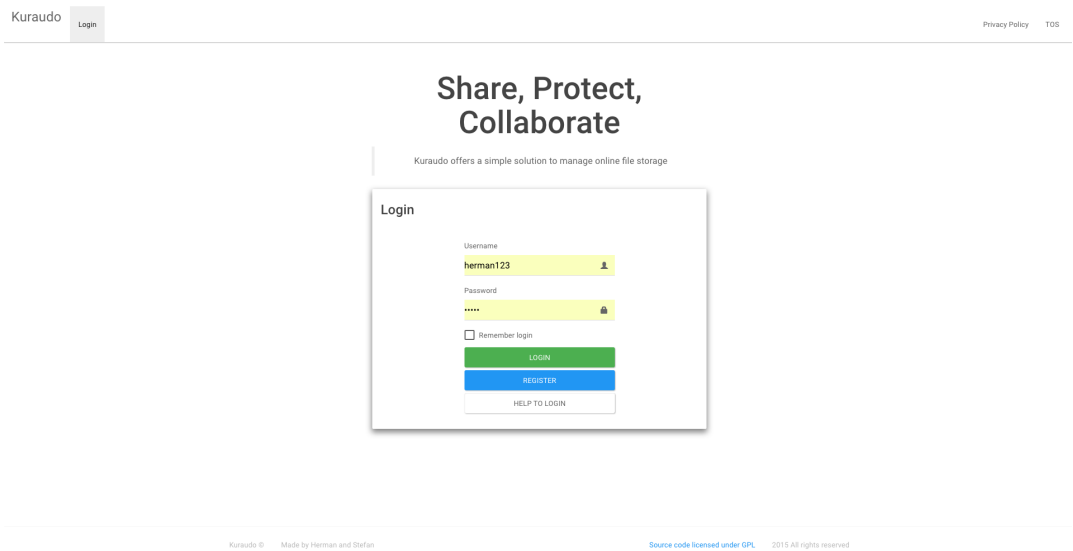


Figure 5: Home page






home			
Name	Kind	Size	Modified
 Developer	Folder		
 Jobs	Folder		
 School	Folder		
 Software	Folder		
 Work	Folder		

Figure 6: Browsing page listing folders

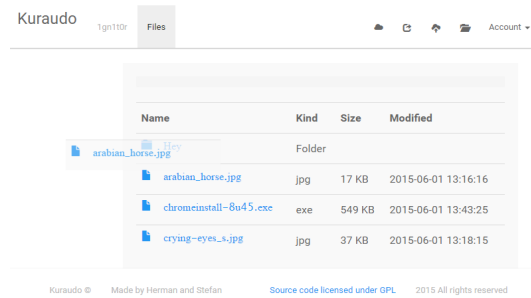


Figure 7: Moving of files

Files and folders may be renamed and deleted by using the right click context menu. Files and folders may be moved by using drag-and-drop functionality. This is shown in figure 7

Multiple files may be uploaded as shown in figure 8. The upload dialog indicates

Folders may be navigated by clicking on them. The breadcrumbs list the current directory and may be used to traverse back to upper directories.

New folders may be created by using the navigation icons (Figure 9). Folders may also be publicly shared or only to certain users using the navigation icons.

Images may be previewed by clicking on image formats.

Public folders are available to non-users of the site through a similar interface (Figure 10).

### 7.3 PROFILE

The profile indicates the user's details such as their name, email and the current user plan they are on. The profile also indicates the space they have available and their past download usage.

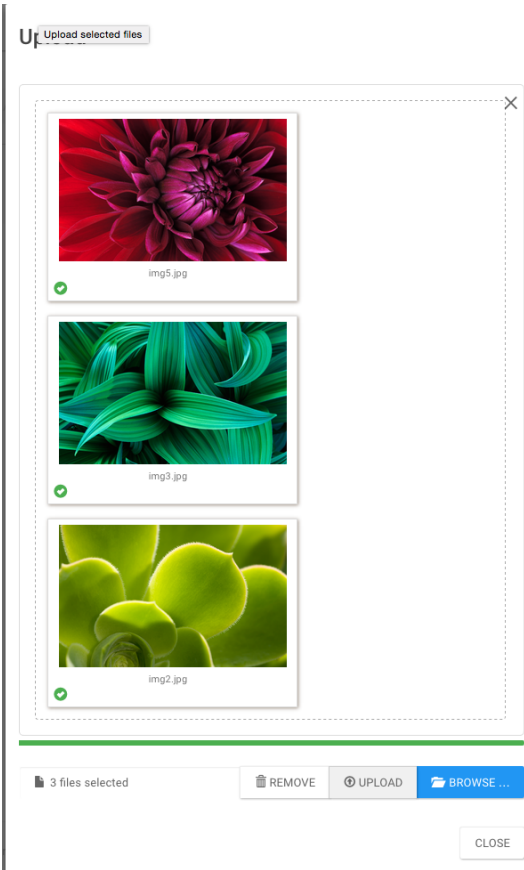


Figure 8: Uploading of files

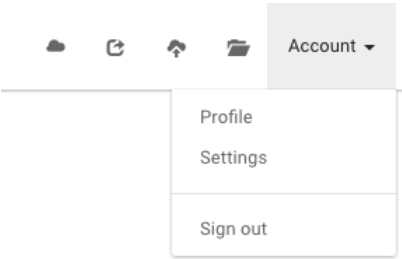






Figure 9: Navigation icons

Kuraudo

Files Public Files

---

29

Name	Kind	Size	Modified
 Hey	Folder		
 arabian_horse.jpg	jpg	17 KB	2015-06-01 13:16:16
 chromeinstall-8u45.exe	exe	549 KB	2015-06-01 13:43:25
 crying-eyes_s.jpg	jpg	37 KB	2015-06-01 13:18:15

Kuraudo © Made by Herman and Stefan [Source code licensed under GPL](#) 2015 All rights reserved

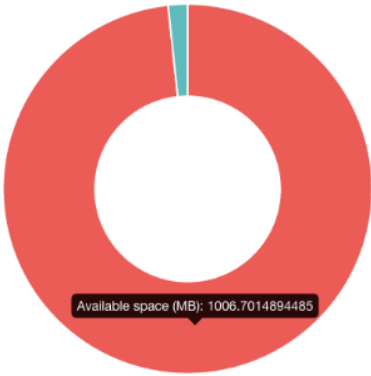
Figure 10: Public folder sharing

## Profile information

Username: herman123  
 First Name: Herman  
 Last Name: Blackie  
 Email: hermanblackie@gmail.com  
 Plan: Free User

## Current Usage

Current space usage



Available space (MB): 1006.7014894485

## Past download usage

by Herman and Stefan

Figure 11: Profile page

## CONCLUSION

---

In this practical the front-end and back-end for a file-sharing website was designed. The database was designed to be in third normal form and features a simplistic design that allows easy file and folder manipulation. In addition it allows only a single unique file to be stored on the hard disk even though the file is uploaded by multiple users.

The data from the database is displayed on a website by using JavaScript, [PHP](#), [CSS](#) and [HTML](#). Bootstrap and JQuery were use improve the development speed. The data from the database is manipulated and retrieved by [PHP](#) calls. These calls are sent through [AJAX](#) in order to keep the site responsive. Using Bootstrap allows the website to be responsive and allows a pleasant viewing experience on a multitude of mobile devices.

The is design follows the Keep it simple, stupid ([KISS](#)) methodology and unneeded complexity was avoided. The site features all of the required specifications in a simplistic and minimal user interface.

## APPENDIX

---

The source code of the site is available at [github.com/1gn1tor/Kurauto](https://github.com/1gn1tor/Kurauto).  
This report is available in pdf at the above link as well.



## BIBLIOGRAPHY

---

- [1] J. A. Hoffer, *Modern database management*. Pearson Education India, 2004.
- [2] B. Prasad, *Concurrent engineering fundamentals*, vol. 1. Prentice Hall Englewood Cliffs, NJ, 1996.
- [3] D. C. Tsichritzis and F. H. Lochovsky, *Data models*. Prentice-Hall, 1982.

## DECLARATION

---

We, Stefan Jacholke and Herman Blackie, hereby verify that this is collaborated work is our own original work.

Whenever contributions of others are involved, every effort was made to indicate this clearly, with due reference to the literature.

No part of this work has been submitted in the past, or is being submitted, for a degree or examination at any other university or course.

*Potchefstroom, June 2, 2015*

---

Stefan Jacholke

---

Herman Blackie