Manchester Metropolitan University

Faculty of Science and Engineering
School of Maths, Computing and Digital Technology

Web Based Personal Organiser

BSc (Hons) Computer Science

Published: 26/03/2018

Author: Joel Cummings
15087385

Supervisor: Dr Kevin Tan

Declaration

No part of this project has been submitted in support of an application for any other degree or qualification at this or any other institute of learning. Apart from those parts of the project containing citations to the work of others, this project is my own unaided work.

27/04/2018

X 

Joel Cummings
Author
Signed by: Joel

Abstract

Today, Personal organisers (also commonly known as personal assistants) have become a very prevalent part of life for the common consumer. These are productivity tools with advanced Personal Information Management (PIM) abilities. Leading software/hardware firms such as Google, Apple, Microsoft, Samsung and others continue to advance and build competing services with the intention of aggressively pushing it into the lives of consumers, through the billions of combined sales of smartphones and other general computing and smart devices to consumers. Enabling this push to offer users with a better way to efficiently work, and live their lives in exchange for allowing one specific firm to manage their data.

Because of constant advancements, personal organisers have become much smarter and therefore a more useful service to users. Many of the top personal organisers today feature similar abilities, meaning no single personal organiser clearly stands above the other as the best to consumers. They are instead differentiated and chosen based on their clear faults and cons, most of which stem from their deep integration into any one ecosystem. This hampers their usability on other platforms which is a problem for consumers who need to use devices from different platforms for their work and personal lives, as it interferes with the flow between the two sides. This means that the user in a usability worst case scenario must be able to either manually transfer data they would like to use, or be able understand small user interface (UI) changes to the app, as it tries to cater to the design language of different platforms as a native application.

3. Design

3.1. Introduction

Following the literature review the purpose of this design specification is to outline the system structure and build upon the requirements of the project, while defining the overall system architecture being mindful of the miss stops existing solutions made, to lead to a successful concept web based personal organiser. This chapter also establishes the application's draft designs of what the intended final application will look like for its first public revision and how it navigates.

3.2. Product Naming

The brand name of any company or product carries a significant amount of weight behind, as it becomes the face of the company/ product which can be directly linked with a positive, or negative connotation for consumers similarly to saying '…judging a book by its cover'. As such, often the name means more than the actual product behind it as they can easily sell products to consumers, that are only aware of the positive connotations of the name that was built up by the company.

Upon envisioning a name for personal organiser that is designed to store all of the user's personal information, the name '**elePersonal**' was settled on as a combination of elephant and personal. This is because elephants have a positive connotation of never forgetting anything, and can be easily turned into a brand logo or mascot; while personal refers back to the personal information management role the web based application provides.

3.3. General Overview & Design Guidelines

Before envisioning the designs for any element that will eventually be used to form the final web application elepersonal, 'Simplicity' was chosen as the core principle for all designs, since the look and feel of any website/application is important (see User Interface Design), however since users of elepersonal aren't visiting the website to evaluate the designs but rather visiting with a purpose; Unnecessary elements cluttering the design of the web application will only serve to hinder the time it takes users to accomplish their purpose, therefore simplicity greatly adds to the usability of the web application. Reading Erik Devaney's (2018) blog on web design, simplicity also goes hand in hand with 'Accessibility' since a simple interface is easier to adapt to a mobile friendly interface,

which is important for the web application to work on mobile platforms as pointed out in the literature review.

Ensuring that simplicity is adhered to as the core design principle, several strategies were set out as guidelines for designing and implementing the system, after performing research on books and short reports such as the "Effective Visual Communication for Graphical User Interfaces" by Suzanne Martin (n.d.).

- Colours: According to the Handbook of Computer-Human Interaction (1997), no greater than five (plus or minus two) colours should be used in the design. Helping users to associate colours with different meanings and to remember them easily. For elepersonal, the colour scheme will be based around the primary colour blue. This is due to the fact that blue can be seen as a relaxing colour because it is everywhere from the sky, to the oceans which is an appealing association to align with elepersonal to make it more attractive visually.
- Typefaces: Typography shouldn't have more than three different typefaces, limited to around three different sizes throughout the website. As such Sans-serif will be the primary font used on elepersonal because it was made to make text displaying on screens clear, as low-resolution screen might lose fine details such "Serifs" at the end of letter strokes.

- Clarity: The meaning of components should not be ambiguous.

- Emphasis: Important components such as navigation, should be distinguishable and easily perceived while non-important components should be de-emphasized to reduce clutter so that more important pieces of information can take center stage as much as possible.

Adhering to these strategies help to reduce time inevitably spent hung up on the aesthetics but also helps keep all designs consistent. Which greatly improves the user experience as there will be a focus for the user's attention, with a combination of layout and colour optimisations will help to focus the user's attention onto important areas such as navigation

components to help the user in finding every bit of information, assisting the user through what is essentially primary and secondary content.

3.3.1. Assumptions, Constraints & Standards

Following on from the Literature survey and comparing competing services, elepersonal has to include certain features on assumption that many of the other bigger services have in common as standard, otherwise, users that are attracted by the clean, simplistic design will end up either moving back to their existing service or moving onto another if they are in search of a service that better fits their needs.

In order to at least match the standards set by other existing solutions, elepersonal at the very minimum has to include several core features found in the literature survey:-

a) Registration and login systems

b) Calendar event management

c) Contact management

d) Notes

e) Tasks

f) Email

These features allow the service to be able to compete with those existing solutions while building on the conclusion within the literature review to build a web based application that expands upon using the internet for personal organisers. Building a comparable web application to that of an application being made native for multiple platforms, also proves beneficial to the applications overall success. As it means that the development of this project is able to disregard limited development man power and/or skill of platform specific developers, allowing the application to run on a wide range of platforms with a relatively fast development time, with no compromises compared to platform specific native applications.

3.4. System Design

Since elepersonal inherently comes with some assumptions about features it will be built with in order to compete with other solutions, the web application will inherently come with some advanced features that some less technology savvy users might be confused about. Referring back to some existing solutions in the literature review, one problem productivity application can face is making themselves a more daunting endeavor for new users, especially with little in-app documentation or tool tips that are quick and easy to understand.

3.4.1. User Interface Design

To combat issues concerning savvy users, elepersonal is being designed to launch with a home section, to give the application a face, somewhere that new users will land on and see when they first visiting the web application. This section of the application is being made to serve several key functions:

1. Advertise the core functionality/features of the application.
2. Offer detailed documentation on each feature, alongside tool tips that will appear in deeper layers of the applications to further help users or quickly remind them what something does.
3. Give the user easy access the help section with common questions about some of the applications features and how they work, before the user must invest, so they know beforehand what they are investing into.
4. Along with the help section, a forum for users to email further queries and feedback to us.

Navigation Design

To keep the design simple, the home section will be primarily navigated using a fixed top navigation bar as seen in Figure 4, which will consist of five self-explanatory buttons. Fours buttons made specifically to serve the four functions covered previously, with a fifth button dedicated to logging the user into the main section.

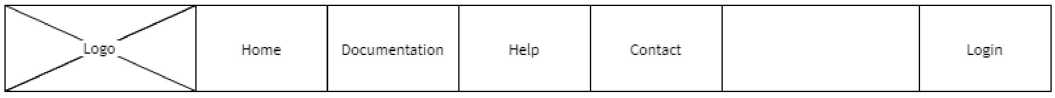


Figure 4: Wireframe design of the main navigation bar.

Using a simple navigation bar that is fixed to the top of the page, also means the design can be easily scaled down for devices with smaller screens as shown in figure 5, to meet the design overview and guidelines.

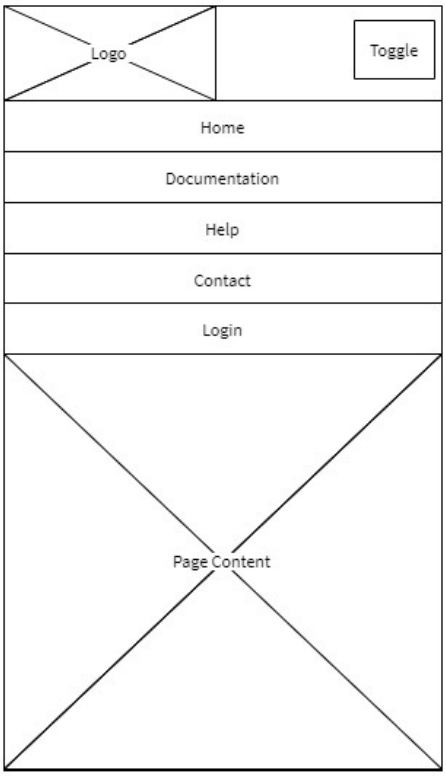


Figure 5: Wireframe design of the main navigation bar for smartphones.

As shown in figure 5, when the web application detects the screen widths has fallen below a specific size, the navigation can be scaled down and remade to a horizontal navigation bar with a new toggle button, which allows for the navigation menu to shown or hidden as needed to allow for other content on the user's screen.

After the user has logged in, they will be able to access to a user profile, settings from a drop down menu as well as the Dashboard for the main web application. From the dashboard they will be able to access every tool featured by this web application, through a second navigation panel that will persist on the left side of the page while the right side will be designed as a dynamic section, shown in figure 6. Meaning that the only changes to the page, will be done within the dynamic section as the user navigates to different pages.

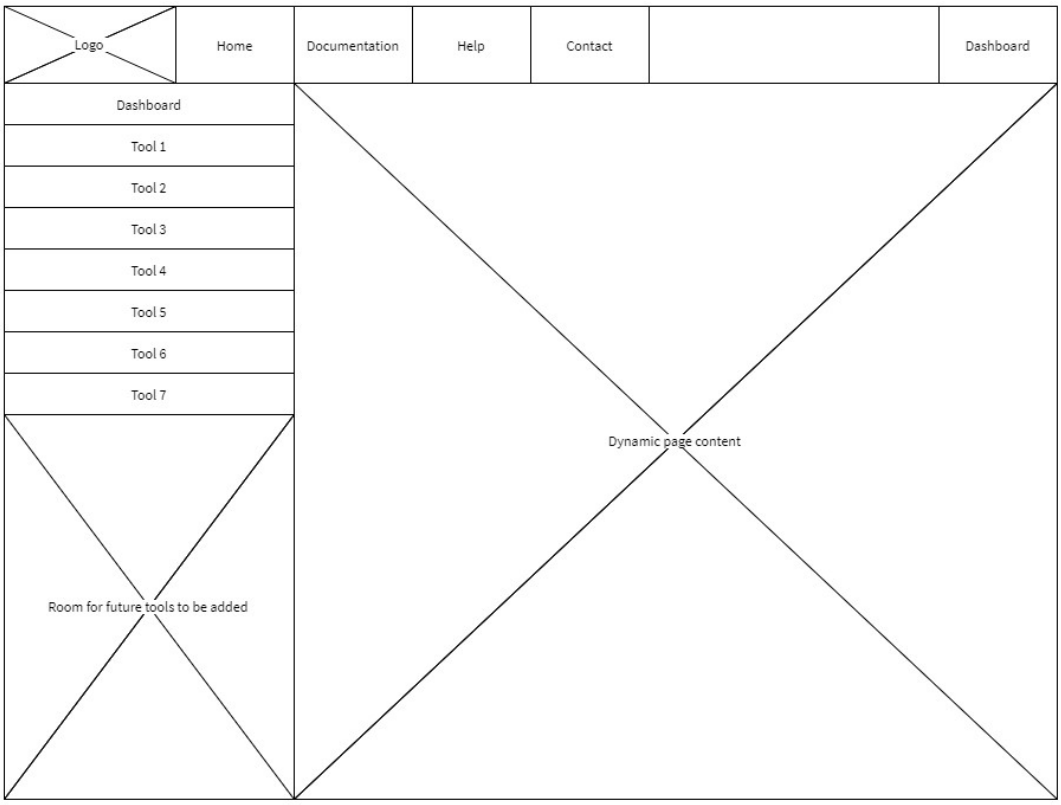


Figure 6: Wireframe design of the main navigation bar for smartphones.

Figure 6 also shows a key aspect of the design that follows simplistic core principles with the side navigation menu on the left-hand side. Although is it simple to have a vertical menu that persists across several pages, this design choice is more future proof compared to others as it allows for further expansion. Meaning that elepersonal's features/ tools are able to grow as the web application matures with future iterations, without a complete redesign to accommodate new additions, while still keeping the design simple and intuitive for new users. Furthermore, keeping the side navigation menu fixed to the left-hand side again means that the design can be scaled down to devices with smaller screens, by turning the menu into a slide out menu as shown in action on figure 7.

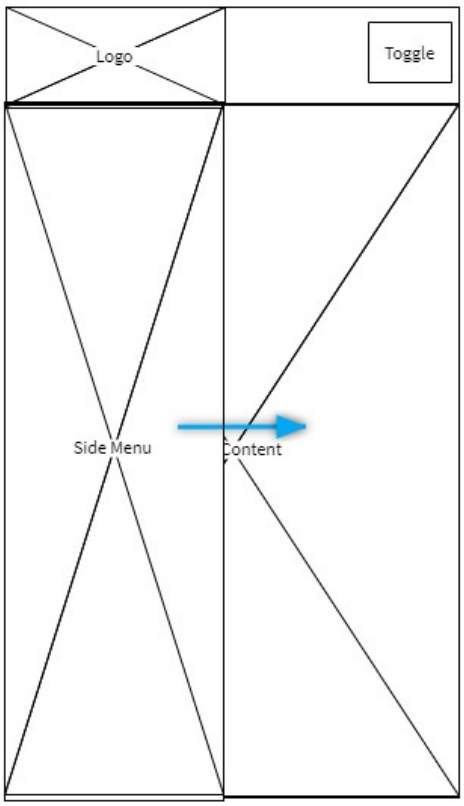


Figure 7: Wireframe design of the side navigation menu sliding out for smartphones.

Dashboard Content Design

The design of the application's user interface (UI) is arguably one of the most important aspects of designing any application or traditional computer program, so comes with a mix of unique challenges associated with it, that directly affect the user's appeal/ interest in the application. As an application cannot for instance:

- Look dated as this will make the application itself seem like a solution/ service for an older generation.
- Look cluttered as this will make the application seem more targeted to professional users and less inviting to potential new users, scaring them away to similar applications that may even be less suited to said user.

As such it is important to have a clean intuitive UI, with a consistent colour theme as well as an intuitive navigation layout. For this reason, the first section of the dashboard will be designed to present the user with widgets, as lite portable extensions of each tool that can live on the dashboard as shown in figure 8. The design choice draws inspiration from Google keep shown in figure 2, because it also allows the user to

quickly glance at a collection of information, presented in an easy to understand format as discussed in the literature review. Furthermore, this design of broken up widgets allows each widget to be realigned into a single vertical layout, as the web application scales down for devices with smaller screens shown in figure 9.

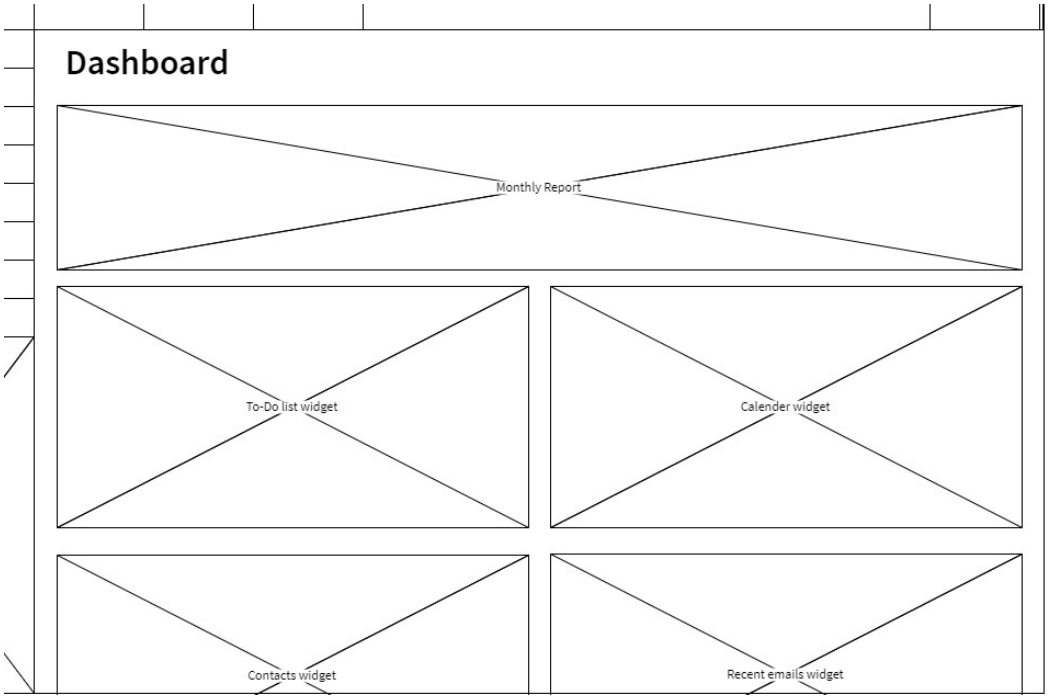


Figure 8: Wireframe design of the dashboard content layout.

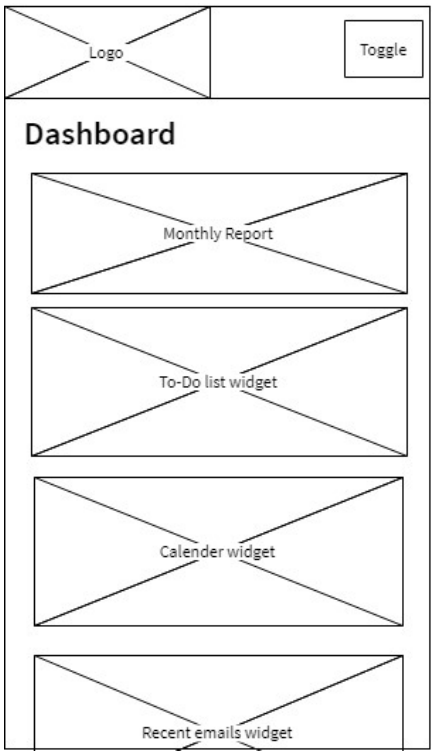


Figure 9: Wireframe design of the dashboard content in a mobile friendly layout.

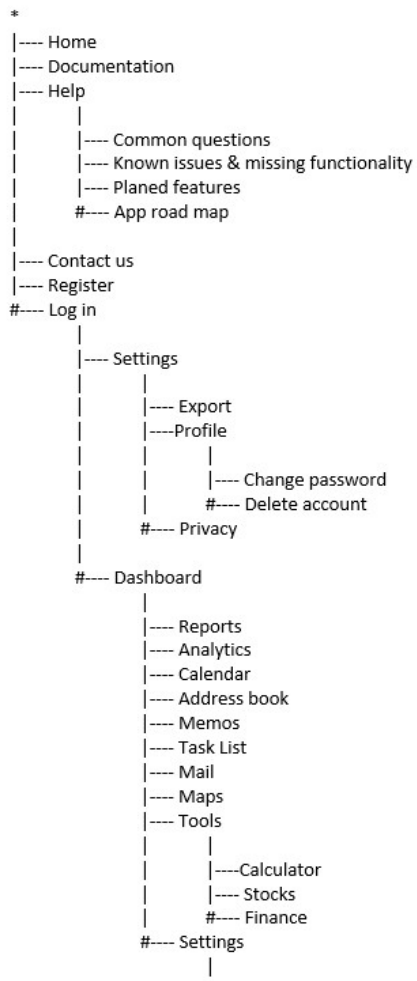
If the user wants access extra functionality related to a specific tool's widget, they are able to navigate to that section of the main tool via a left side menu, or by clicking on the related widget as shown in figure 8.

3.5. Architecture Design

Following a top-down approach to designing the entire system, breaking down the different elements of the system. Allows the sub-systems to be better designed to accommodate for the other systems with the insights gained by this approach.

3.5.1. Directory structure

After setting out the design elements for the user interface, the directory structure of the application for user navigation will follow a specific structure showing some use cases.



Focusing on the ‘Log in’ element of the directory, the web application will be intended to dynamically switch the login and registration buttons out for different buttons, that will allow the user direct access to the dashboard their profile and settings after they have successfully logged in, since the login and registration buttons are made redundant.

3.5.2. Software Architecture

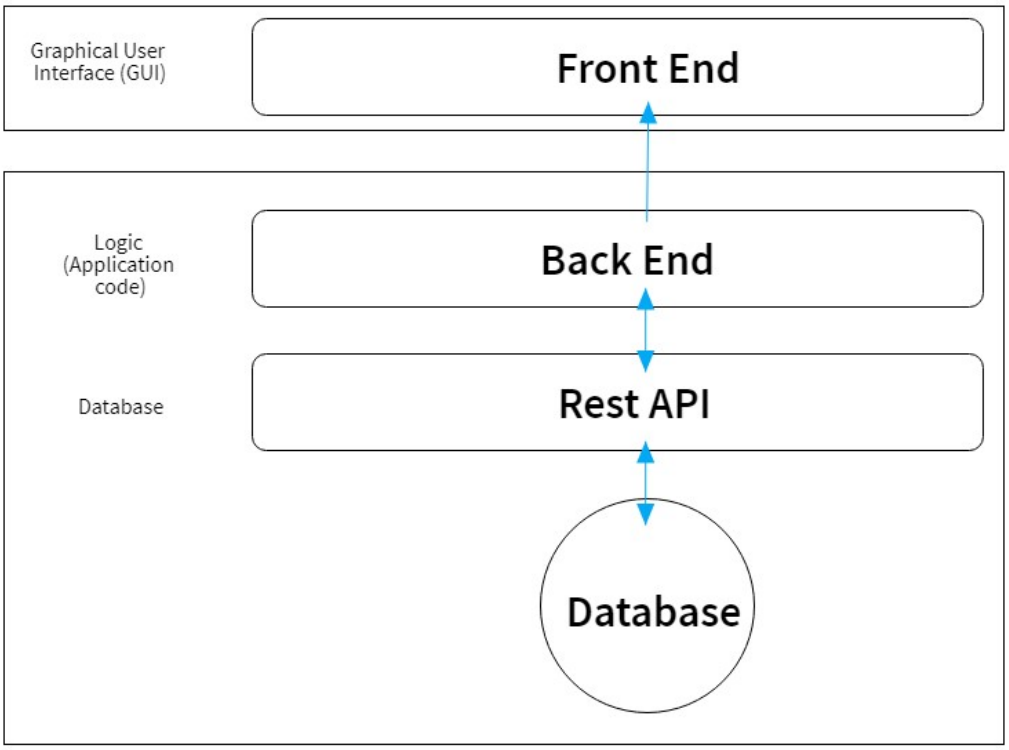


Figure 10: Software Architecture.

Figure 10 shows a visual representation of the overall architecture of the project. Broken up into two sections, the web based application is design to be ran from a server where HTTP requests can be handled to manipulate a database, to query and return information for the user.

3.5.3. Database Designs

Figure 11 shows the design implementation if the tables within the database back end, as well as showing how they are linked together.

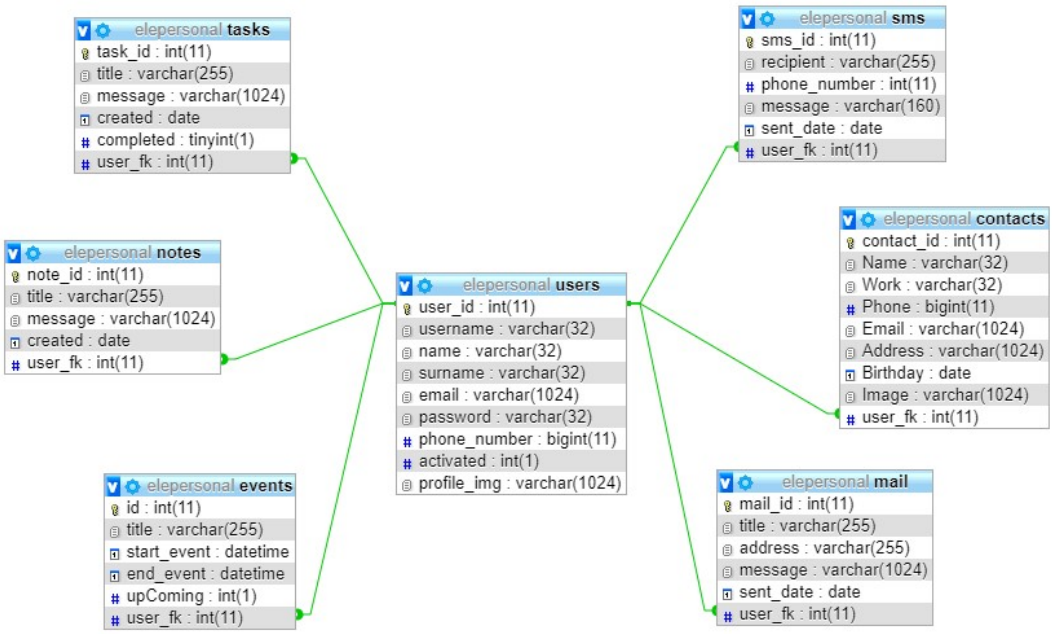


Figure 11: Database Entity Relationship Diagram

4. Implementation

4.1. Introduction

Building upon the design foundations that were outlined within the design chapter, a focus on programming languages and software tools can be expanded on within this chapter, as the creation of elepersonal is outlined. This chapter also explains some of the challenges faced with creating a personal organiser from scratch.

4.2. Programming Language

As the outlined goal for creating a personal organiser is to make the application web based, in order expand upon the advantages that affords as a solution to a more ubiquitous computing vision discussed in the literature review. The primary programming language will consist of Hypertext Markup Language 5 (or more commonly referred to as HTML 5) for creating the application as a web app, along with Cascading Style Sheets (CSS) and JavaScript which together form the defining technologies behind the modern world wide web (WWW.) that all browsers today recognise. But when Referring to the figure 10 visual representation of the software architecture, it is clear that the scope of this project also requires the use of domain-specific language such as structured query language (SQL), to manage data stored within a relational database management system (RDBMS) such as MySQL which is an open source RDBMS, that has all of the capabilities needed by such a project.

However, HTML was not designed to communicate with SQL database management systems, so sever-side scripting language such as PHP (recursive acronym for Hypertext Preprocessor) which has some components of C/C++, has to be embedded into the HTML code to expand its functionality as seen in figure 12.

```
<div class="col-sm-12">
    <h1>Recover Username</h1>
    <!--Check if the user recently made a change to their password by checking the URL,
    to either display success message or trigger the change password function -->
    <?php
    if (isset($_GET['success']) && empty($_GET['success'])){
        echo 'You have successfully changed passwords';
    } else {
        if (empty($_POST) === false && empty($errors) === true) {
            //posted the form and no errors
            change_password($session_user_id, $_POST['password1']);
            header( string: 'Location: changePassword.php?success');
        } else if (empty($errors) === false) {
            //output
            echo outputErrors($errors);
        }
    }

    ?>
</div>
```

Figure 12: Sample of Php code embedded within HTML code block.

4.3. XAMPP

A web based application as the name implies, has to live on a server where the user's browser is able to access and pull the necessary resources anywhere in the world over the world wide web (WWW). However, without a private local server available on which to build the personal organiser, and renting a server would be impractical financially and for rapid testing. The open-source solutions known as XAMPP was employed. The development tool Allows developers to build and test projects quickly on local computers without access to the internet, as a local (on machine) web server software stack. From the software stack provided by XAMPP as shown in figure 13, the software components needed were the Apache HTTP web server in order to host, the MariaDB relational database management system (RDBMS) that is also provided in the software stack, along with the interpreters needed to execute the PHP scripts due to PHP not natively being supported by web browsers.

Figure 13: XAMPP software stack control panel.

When Apache is running, it creates a local IP Address as a domain, that can be reached through the browser if the address 127.0.0.1, or localhost is entered into the browser's URL as seen in figure 14.

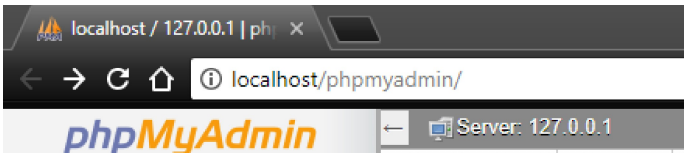


Figure 14: Browser accessing Apache HTTP web server domain

4.3.1. MySQL

Running MariaDB (MySQL fork) alongside Apache, allows for access to the database administration tool 'phpMyAdmin' seen in figure 14. With phpMyAdmin, the database tables outlined in the database design along with any table relationships, are able to be created (see figure 13) for manipulation by the personal organiser as it executes queries that insert, update or delete data in any table on the database.

	Table	Action	Rows	Type
☐	contacts	Browse Structure Search Insert Empty Drop	6	InnoDB
☐	events	Browse Structure Search Insert Empty Drop	4	InnoDB
☐	mail	Browse Structure Search Insert Empty Drop	0	InnoDB
☐	notes	Browse Structure Search Insert Empty Drop	9	InnoDB
☐	sms	Browse Structure Search Insert Empty Drop	0	InnoDB
☐	tasks	Browse Structure Search Insert Empty Drop	5	InnoDB
☐	users	Browse Structure Search Insert Empty Drop	6	InnoDB
	7 tables	Sum	30	InnoDB

Figure 15: Database tables in phpMyAdmin.

4.4. PhpStorm

In order to build the web application after the database backend has been created in phpMyAdmin, work on building the web application needs to be done within an editor that provides support for PHP, than a standard editor that only supports HTML, CSS and JavaScript because any PHP scripts have to be interpreted.

PhpStorm is a commercial integrated development environment (IDE) for PHP, with full support for HTML, CSS and JavaScript (shown in figure 16 & 17) with useful features such as code analysis and completion. However, for students there is a 1 year free license that allowed PhpStorm to be the clear choice to use for implementing the web application design, as the editor is the most fully featured.

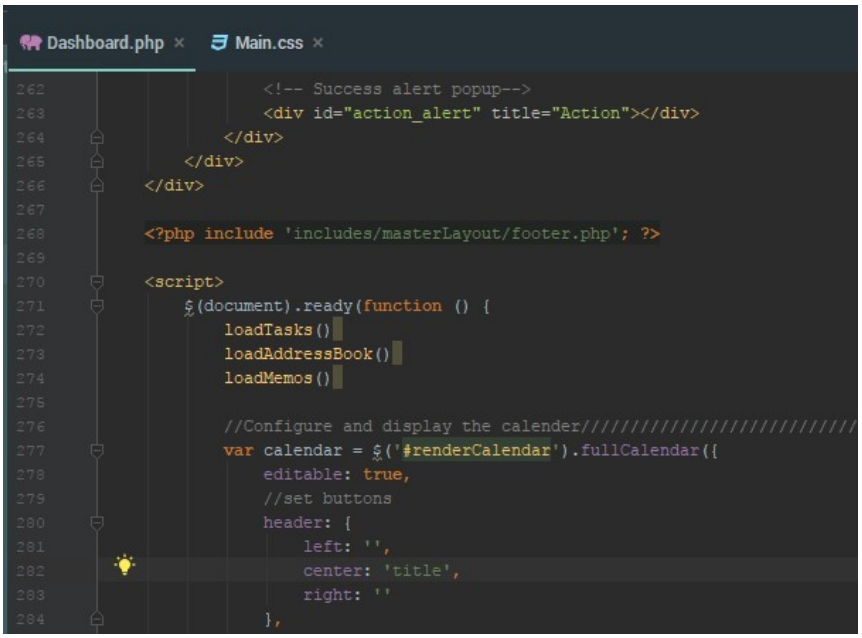


Figure 16: Segment of HTML, PHP and JavaScript code in PhpStorm.

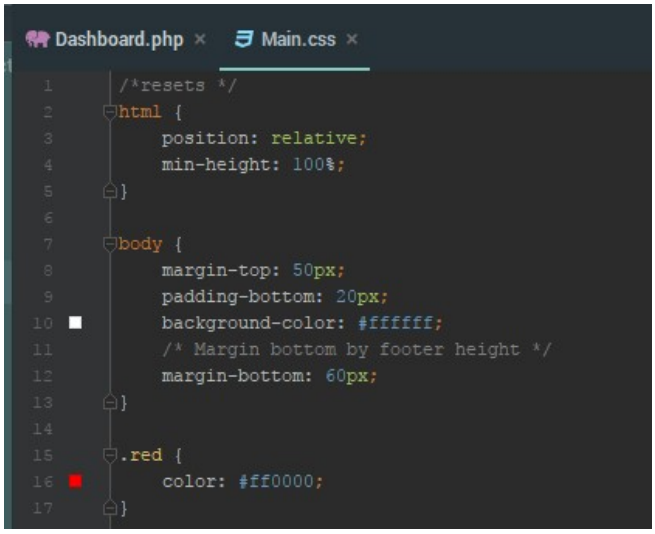


Figure 17: CSS code in PhpStorm.

4.5. Feature Implementation

Before building on the design structure laid out in the design brief for the personal organiser, the core systems: -

a) Registration and login system

b) Calendar event management system

c) Contact management system

d) Notes system

e) Tasks system

outlined in the 'Assumptions, Constraints & Standards' have to be implemented.

4.5.1. Registration & Login System

The first system of the personal organiser that users will interact with is the registration and login and system. Identified as one system because they both execute queries to the user table with the database; these two components are the first entities that users interact with depending if they are a new or existing user.

Registration component

Designed similarly to a setup wizard for new software installations, the registration component consists of a form that is split into 3 sections. This allows the user to be walked through which pieces of personal information is required, in order to make an account for the personal organiser. Shown with figure 18 and 18, the wizard shows which step of the registration the user is on, along with buttons to progress with the registration or to cancel and return to the home page.

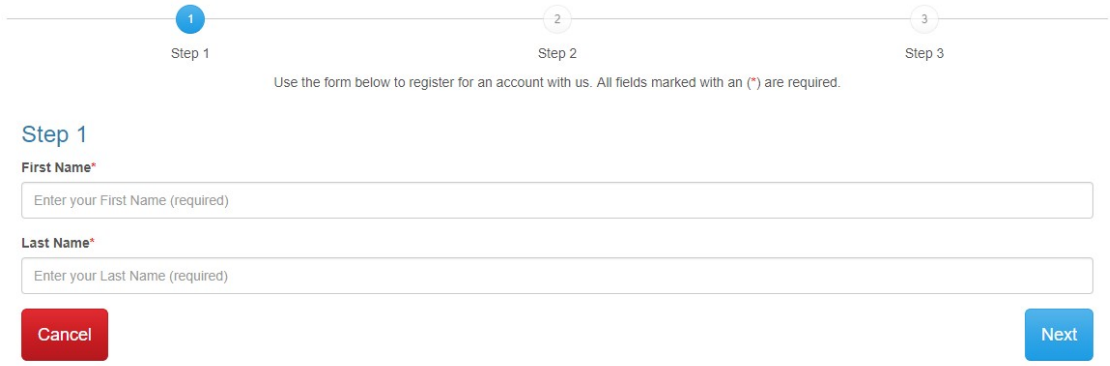


Figure 18: First step of the registration wizard

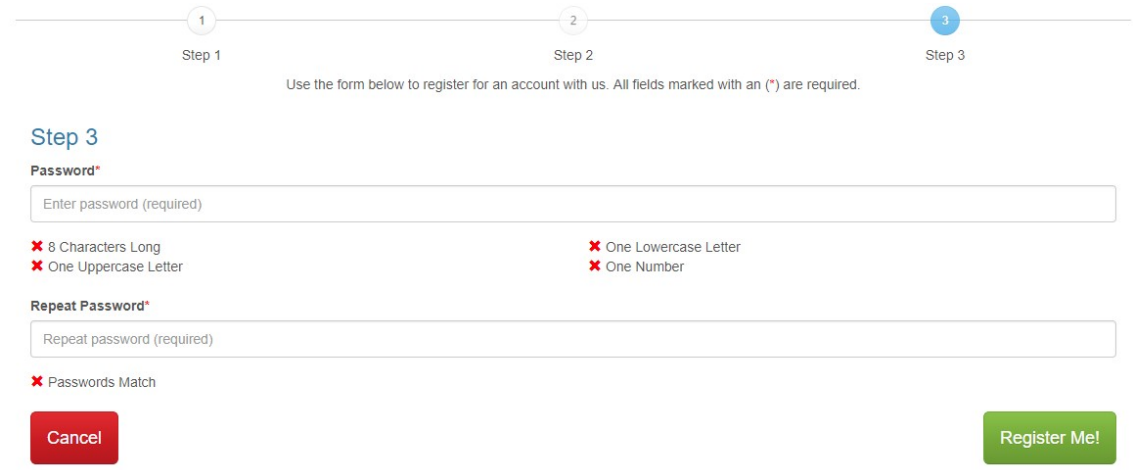


Figure 19: Last step of the registration system

Once the user reaches the last step of the wizard and clicks on the 'Register Me!' button shown in figure 19, the form is submitted for error checking and MySQL injection attack protection before executing an insert query, that will add the user's data into the users table as shown in figures 20-22.

```php
//check that the form has been submitted and validate form data
if (empty($_POST) === false) {
    $required_fields = array('name', 'surname', 'username', 'email', 'password', 'rpassword');

    foreach ($_POST as $key => $value) {
        //if a value that has been set to required is empty
        if (empty($value) && in_array($key, $required_fields) === true) {
            $errors[] = 'Highlighted fields are required';
            //exit loop if 1 error is found
            break 1;
        }
    }

    //If no error with required fields, check individual fields for further errors.
    if (empty($errors) === true) {
        //stop existing user account being made again
        if (user_exists($_POST['username']) === true) {
            $errors[] = 'Username already taken!';
        }
```

Figure 20: Error checking

```php
//filter string to protect against sql injection attack
function sanitize($data)
{
    //htmlentities to help again user entered scripts by stripping tags and
    //encoding strings to escaped SQL string
    return htmlentities(strip_tags(mysqli_real_escape_string($GLOBALS['conn'], $data)));
}

function array_sanitize(&$item)
{
    //htmlentities to help again user entered scripts by stripping tags and
    //encoding strings to escaped SQL string
    $item = htmlentities(strip_tags(mysqli_real_escape_string($GLOBALS['conn'], $item)));
}
```

Figure 21: MySQL injection protection

```php
function register_user($register_data)
{
    //Make sure all strings are legal for SQL statement
    array_walk( &array: $register_data,  funcname: 'array_sanitize');
    //Encrypt password
    $register_data['password'] = md5($register_data['password']);

    //Format strings in standard SQL format for execution
    $fields = implode( glue: ', ', array_keys($register_data));
    $data = '\'' . implode( glue: '\', \'', $register_data) . '\'';
    $query = "INSERT INTO `users` ($fields) VALUES ($data)";
    debug_to_console($query); //Check output

    //Execute query on database
    mysqli_query($GLOBALS['conn'], $query);

}
```

Figure 22: Function to create query and execute in database.

Login Component

Instead of requiring an entirely separate page for a simple login component that would waste a lot of screen space, the login component was made as a simple dropdown panel/ widget of the navigation bar, to gives the user quick access as seen in figure 23.

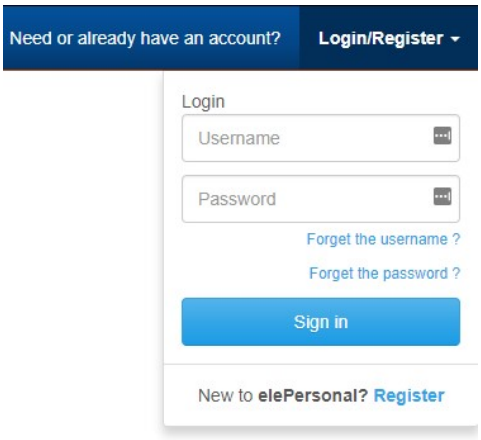


Figure 23: Dropdown login/ register panel

The back end code for the login component is also relatively simple. This is because other than more error checking similar to figure 20, the login component only needs to retrieve the user_id from the users table as shown in figure 24. With the user_id, the session id can be set which the application will see to give the user access to their account. See figure 24.

```php
//Check if session has been set, to allow logged in
function logged_in()
{
    return (isset($_SESSION['user_id'])) ? true : false;
}
```

Figure 24: Has the session been set.

For example, the navigation bar in figure 25 checks the logged_in function that is shown in figure 24, to decide whether to include the dropdown login/registration panel button or the dashboard and user profile buttons.

```php
<ul class="nav navbar-nav">
    <li class="nav-item"><a href="Index.php">Home</a></li>
    <li class="nav-item"><a href="Documentation.php">Documentation</a></li>
    <li class="nav-item"><a href="Help.php">Help</a></li>
    <li class="nav-item"><a href="Contact.php">Contact</a></li>
</ul>
<ul class="nav navbar-nav navbar-right">

    <?php
    if (logged_in() === true) {
        include('includes/widgets/logout.php');
    } else {
        include('includes/widgets/login.php');
    }

    ?>
</ul>
```

Figure 25: Navigation bar logic to switch buttons.

4.5.2. Calendar Event Management System

Unlike the other systems implemented into the personal organiser, the calendar event management system employs the FullCalendar plugin (figure 26) in order to display a drag-n-drop event calendar for testing. This was used due to time constraints, as developing a full-size functioning calendar that properly scale to a variety of sizes for other platforms, could be an entire project in itself.

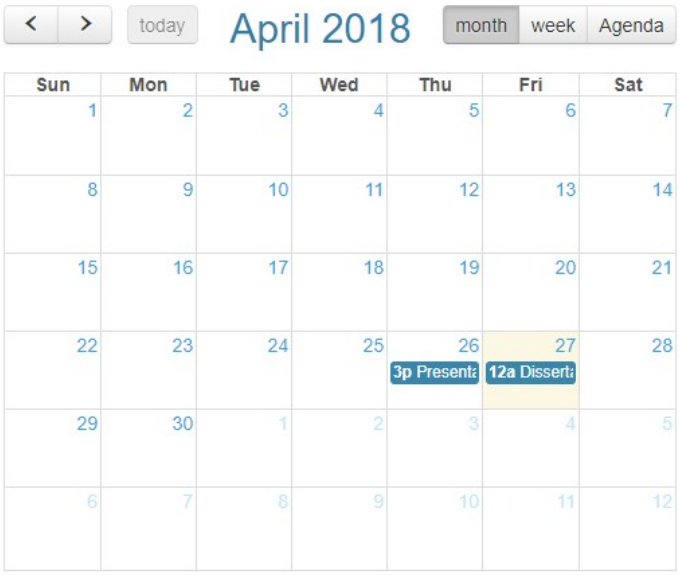


Figure 26: FullCalendar

However, writing the scripts (shown in figure 27) for the plugin allows various visual parameters to be adjusted, and for the calendar to be configured for executing queries on the events table within the local database.

```
defaultView: 'month',
navLinks: true, //allow day/week name to navigate view
eventLimit: true,
//load data
events: 'core/functions/calendar_load.php',
//make cells selectable
selectable: true,
selectHelper: true,

//insert event for selected cell
select: function (start, end, allDay) {
    var title = prompt("Enter Event Title");
    if (title) {
        //get current date and time
        var start = $.fullCalendar.formatDate(start, "Y-MM-DD HH:mm:ss");
        var end = $.fullCalendar.formatDate(end, "Y-MM-DD HH:mm:ss");
        //inset the event
        $.ajax({
            url: "core/functions/calendar_insert.php",
            type: "POST",
            data: {title: title, start: start, end: end},
            success: function () {
                //reload event data
                calendar.fullCalendar('refetchEvents');
                alert("Added Successfully");
            }
        })
    }
},
```

Figure 27: Section of FullCalendar script

4.5.3. Contact, Notes & Tasks Systems

Although the contact, notes and tasks systems are all different in terms of which table they manipulate within the database, each of the systems relatively have the same design and code structure. As such components outline for one within this section, is applicable to the other two systems.

Focusing on the contacts management component of the personal organiser, as shown in figure 28. The system is designed to show the user a list of contacts they have stored in the database, under their specific user ID. Allow the user to add, view, update and delete contacts only they have made.

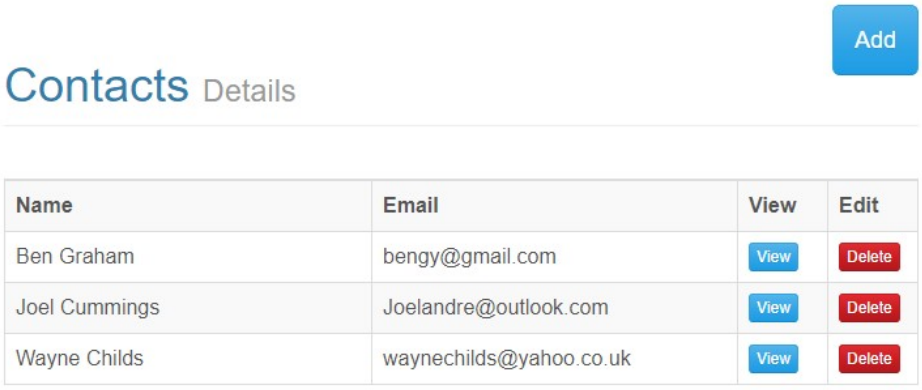


Figure 28: Contact Management System

4.6. Challenges

Most of the challenges associated with developing a web application, comes down to experience with the different languages. In this case, particularly with learning best practises with writing JavaScript's as shown in figure 27.

Timetable and Deliverables

Based on the deliverables deadline of the previous year, my timetable for deliverables will look something like this project planner.

Project Planner

Select a period to highlight at right. A legend describing the charting follows.

Period Highlight: 40

Plan Duration | Actual Start | % Complete | Actual (beyond plan) | % Complete (beyond plan)

ACTIVITY	PLAN START (week)	PLAN DURATION (weeks)	ACTUAL START (week)	ACTUAL DURATION (weeks)	PERCENT COMPLETE
Project Duration	0	1	1	25	0%
Terms of Reference	1	1	1	1	100%
Ethics Form submitted	1	1	1	1	100%
Literature Review	4	6	0	1	0%
Product Design	7	6	0	1	0%
Evaluation Design	11	6	0	1	0%
Report Structure	14	6	0	1	0%
Draft Slides	16	6	0	1	0%
Practice presentation	20	3	0	1	0%
Final Report	16	10	0	1	0%
Final Product	11	15	0	1	0%

PERIODS 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35