



STUDENT ID	16901284
LOGIN ID	ak750
SUBMISSION YEAR	2017
WORD COUNT	10,900

Masters in Computer Science

AUTHENTICATION SERVICE PROVIDER

AUTHOR
ARJUN KRISHNA PRASAD

SUPERVISOR
RADU GRIGORE

Abstract

With numerous start-ups appearing around the globe and so are their respective websites, the privacy of information is being evaded by hackers on a regular basis. Hence, there is a need for a small and medium sized companies to use authentication system for websites restricted to their employees. Therefore, a need for guarded and secure login system. Unfortunately, this means extra bucks for the start-ups to procure the technology, which can be a luxury for many. This paper is about free-of-cost web service named 'Dot.Connect' that integrates secure features of a login system by including plausible and reliable validation and verification. Basically, acting as an authentication service provider. The feature can be utilized by any registered employee of a company. The website has been coupled with attractive UI design with some study on color scheme and web design features.

Acknowledgements

Sincere thanks of gratitude to my supervisor, Radu Grigore, Lecturer at School of Computing, University of Kent, who gave me this opportunity to develop the web feature of my choice, and also for guiding and supervising me from inception till end. His inputs for the project have been of great help. The idea of including the company registration and employee registration process was his, without which the project would have been incomplete.

The dissertation is a tribute to my Dad and Mom for their unconditional love and support, and for believing in my abilities.

Contents

Abstract	i
Acknowledgements	ii
List of Figures	v
Abbreviations	vii
1 Introduction	1
1.1 Objective	1
1.2 Paper Details	2
2 Background	3
2.1 Security Related Article Review	4
3 Project Foundation	7
3.1 Company Registration	7
3.2 Employee Registration	8
3.3 Login System	9
4 Methodology & Implementation	12
4.1 Database Structure	12
4.2 Website Logo & Design	16
4.3 Implementation of Web-Design	17
4.4 Company Registration	20
4.5 Employee Registration	23
4.6 Authentication or Login System	26
5 Precautionary & Security Measures	31
5.1 Web Vulnerabilities	31
5.1.1 Cross Site Scripting (XSS)	31
5.1.2 SQL Injection	34
5.1.3 Brute Force and Dictionary attack	36
5.2 Measures Taken	37
5.2.1 REGEX Validation and other checks	37
5.2.2 Sanitization	39
5.2.3 Prepared Statements	40

5.2.4	Conditions for setting Passwords	41
5.2.5	Minimum Password Attempt	43
5.2.6	Hashing & Salting (cryptography)	43
5.2.7	Usage of Tokens	44
6	Testing & Scrutiny	46
6.1	Black Box Testing	46
6.2	Post-Mortem	47
6.2.1	Password Conditions	47
6.2.2	Company Website Specification	48
6.2.3	Data transfer	48
6.2.4	Company Detail Updation	49
6.2.5	Point Of Contact update	49
6.2.6	No Freemails rule - not rigidly enforced	50
6.2.7	Unit Testing not done	51
6.2.8	Cohesion and Coupling issues	51
7	Conclusion	52
7.1	Future Developments	52
7.1.1	Structural Development	52
7.1.2	Service on Mobile Platform	53
7.1.3	Better use of Data Collection	53
7.1.4	reCAPTCHA	53
7.1.5	Extra Features	53
7.1.6	Customer Feedback	54
A	Black Box Test Results - Company Registration	55
B	Black Box Test Results - Company Account Activation	57
C	Black Box Test Results - Company Details Updation	58
D	Black Box Test Results - Employee Registration	60
E	Black Box Test Results - Employee Account Activation	64
F	Black Box Test Results - Employee Login	65
G	Black Box Test Results - Employee Password Reset	67
H	Choice of Programming Language	68
	Bibliography	70

List of Figures

3.1	Company Registration.	8
3.2	Employee Registration.	9
3.3	Login System.	10
4.1	Conceptual Model: UML Diagram.	13
4.2	SQL database	15
4.3	Dot.Connect Logo Design	16
4.4	Login System element design	17
4.5	Website Design & Composition	18
4.6	Company Registration attributes	21
4.7	Alert Message	21
4.8	Company BRAM Films with active attribute set to 0	22
4.9	Company BRAM Films with active attribute set to 1	22
4.10	Employee Registration attributes	24
4.11	Employee Detail storage in database	24
4.12	Location Table detail storage in database	25
4.13	Mobile Table detail storage in database	25
4.14	Token for mobile verification through SMS	25
4.15	Login System attributes	27
4.16	Error Message for company not registered	28
4.17	Error Message for employee account not registered	28
4.18	Error Message for employee account not being active	28
4.19	Error Message for not matching credentials	29
4.20	Error Message for account block	29
5.1	Input: Normal Message	32
5.2	Output: Normal Message	32
5.3	Input: Message with HTML tags	33
5.4	Output: Message with HTML tags	33
5.5	Input: Message with some malicious code	33
5.6	Output: Message with some malicious code leading to alert	34
5.7	Conditions for setting Password	41
5.8	Database depicting storage of hash and salt	44
5.9	Email for account activation	45
6.1	Company Registration Test: Assume Registration Number is unique	47
6.2	Company Registration Test: Invalid Company Title	47
6.3	Company Registration Test: Blank First Name	47

6.4	Company Registration: Point of Contact input in 2 separate entity . . .	49
6.5	Company Detail Updation: Point of Contact input as single entity . . .	49
6.6	Suggestion Box for emails during company registration	50
6.7	Test result when using Gmail as Official mail ID	50
6.8	Test result when using AOL freemail as Official mail ID	50

Abbreviations

ADs	AD vertisements
API	A pplication P rogramming I nterface
BSD	B erkeley S oftware D istribution
CSS	C ascading S tyle S heets
FK	F oreign K ey
GPU	G raphics P rocessing U nit
HTML	H yper T ext M arkup L anguage
IP Address	I nternet P rotocol A ddress
MD5	M essage D igest A lgorithm 5
PDO	P HP D ata O bjects
PHP	H ypertext P re P rocessor
PK	P rimary K ey
SHA	S ecure H ash A lgorithm
SMS	S hort M essage S ervice
SQL	S tructured Q uery L anguage
SQLIA	S tructured Q uery L anguage I njection A ttack
SQLIV	S tructured Q uery L anguage I njection V ulnerability
UI	U ser I nterface
UML	U nified A Modelling L anguage
URL	U niform R esource L ocator
US	U nited S tates
VPN	V irtual P rivate N etwork
XAMMP	C ross(X)- P latform A pache M ySQL P HP P erl
XSS	X -cross- S ite S cripting

Chapter 1

Introduction

The world today is divided between people who are optimistic and people who are pessimistic. The community that deals with information security, are always the later and rightly so. It is said that the security is an illusion. There is hardly any piece of information that cannot be possibly stolen. There is absolutely no data that cant be made incompetent by change. Having said that, security is an important factor, keeping in mind, it needs to be regularly updated. It still is a useful device that has to be put into place to ward off unauthorized specimen from extracting the information. It is even more significant, the kind of security preferred, as that can have long lasting effect on the safety of the information.

When it comes to web security, it may be debateable to assume that the data may or may not be worthy enough to be hacked, however, website details are compromised all the time, even if hackers just want to test his new found skills on a worthless piece of data.

1.1 Objective

The objective of the project is to lend a helping hand to the tiny companies like the start-ups financially, by making a cost-free system to restrict unapproved users from accessing the information.

The authentication service or the login system provides certainty of safety. But let us be clear about one thing – a login system is not the only measure that needs to be

taken to keep the data safe, but it is effective enough to keep most unwarranted users from extracting important details, or to manipulate them.

1.2 Paper Details

This paper details the technology used, the methods adopted and the designs implemented in creating an Authentication Service Provider, and analysis on the final outcome of the service. Explanation on how the authentication provider works towards preventing hacking on such login systems is provided and also provide inputs on advantages and disadvantages of techniques deployed to operate. Apart from this, explanation on what is lacking in the system built and what are the improvements that can be done in the future has been discussed .

The UI design, for example, the color scheme and the web composition, and the technical enhancement of user experience has been studied in detail.

Chapter 2

Background

The concept of Federated IDs have been there for a while now. It is the means of linking a person's electronic identity and attributes, stored across multiple distinct identity management systems.[1] Suppose an app needs to create a login system for its users, they could simply use the federated ID management system of popular websites which are already managing millions of users like Twitter[2], Facebook[3] or even Google[4], who have made their APIs available on their developer tools. This way, they are obtaining a system to authenticate their users and also keep the security of the user data guaranteed.

This concept can be used by businesses around the world as a login system for their website, so as to enable their employees to login to their system. The identity of their employees and how to allow only those users to login, are issues that can be managed easily.

But on the flip-side, there is a drawback to this approach. Once the business purchases a domain address, the employees are eligible for their own unique email IDs and password, but these credentials cannot be used for websites with federated authentication system as the employees are supposed to have a separate account on Twitter or any website they have chosen to manage their IDs, which is considered an unprofessional approach.

This is where our project comes into light as an alternative. The web service, is business specific and use the employees official email IDs as credentials to login. Apart from this, the project could also be further developed, customized to company specific.

As the website is supposed to manage data and accounts of numerous individuals from numerous companies, security was a priority. Our biggest challenge was to manage them as per the modern security requirements.

2.1 Security Related Article Review

When it is targeted on vulnerable sites, SQL Injection Attack (SQLIA) is one of the most dangerous form of hacking to have plagued web applications for years. It is one of the numerous ways budding hackers take advantage of online vulnerability because of the ease in which it can be performed. Experts over the years have come out with several ways in which these type of attacks can be guarded against. A publication named Defeating SQL Injection[5] by Lwin Khin Shar and Hee Beng Kuan Tan speaks about some of the best practices for countering SQLIA.

SQLIA is a technique of hacking whereby the hacker passes SQL commands as input values in order to manipulate inner codes of the SQL statement in the database, to the hackers advantage. As per the authors, it is mainly caused by the limited amount of validation and sanitization of the user inputs like the absence of checks, and also due to the weak handling of the methods by the developers.

The PHP function `mysql_real_escape_string()` has the ability to act as a barrier to any enforcement of database operators such as LIKE, GRANT but does not deter wildcard characters. Hackers can easily encroach by using % and _ in the password. This helps in matching more passwords, thereby seeping through barriers created. The failure to have proper data checks in place, can also be an issue. Another complication the program can run into is when the attacker provides HEX code to the parameter name and the parser, which can be converted into varchar value in the database. They further touch upon improper parameterized binding as well.

The authors have proposed an effective three-pronged approach to deal with the menace. They have suggested Defensive Coding approach as the first fortification and necessary tool. This can be incorporated with SQL Injection Vulnerability (SQLIV) detection, and SQLIA runtime prevention.

Defensive coding is uncomplicated, easier approach. For the technique to be applied, the authors have recommended several manual defensive coding checks to combat threat. Parametrized queries are an important part of the procedure. This forces the developers to first enforce the query before passing any values. This way any extra commands injection is avoided from the user end. If dynamic query is a necessity, escaping every inputs from the user, is a better option. Apart from these, Data Type Validation is also touched upon. This ensures the procedure deters any inconsistency in the user defined inputs. White list filtering against black list filtering has been recommended by the authors for inputs such as email IDs, dates, etc.

The authors caution that manual defensive technique can be error-prone and also the programmer might have to spend a lot of time if the user inputs are manifold. To counter this, SQL DOM, an automated testing technique could also be used, though the earlier technique is far better. Parameterized query insertion function `mysql_query(enter SQL query here)` is another way of dealing the issue, which automatically handles the statement by binding the data passed in the prepared statements. The downside to this is that, it only works with explicit strings in SQL.

SQLIV detection is the 2nd barricade that the authors advocates. They have suggested 4 different types of detection that can be used, each with its own pros and cons. Code based vulnerability testing, which is efficient in recognizing any vulnerabilities in place is one way of testing. But this approach does not directly find problems as it needs to be assessed by the programmer by passing inputs.

Concrete attack generation is a technique that generates set of inputs already in place which can automatically detect SQLIVs unlike the previous one, when provided with path to be exercised. This is known as symbolic execution. This technique could also be used to enforce manual data to check if it seeps through the barriers. The authors talk about constraint solvers like SUSHI, Andrilla, and SWAT, which automatically test the programs for vulnerabilities. But having said that, they warn us on the shortcomings because effective automated testing unit have not been developed yet.

Another SQLIV detection method is Taint-based. In this, the user inputs are usually isolated so that the harmful inputs do not infect the rest of the data. Many analysis can be applied to identify the user input and they are made sure its sanitized and validated. The issue with this approach is, if the inputs are passed through libraries

and function, it might fail to identify. The final detection method is Data-mining based but authors warn that this type of analysis is not accurate.

Runtime SQLIV prevention is the 3rd and final block for SQLIA. But the downside to this approach is it can make the finding of bugs for vulnerabilities very complicated. Out of the many techniques the author talks about, they have elaborated on Learning based prevention technique where static and dynamic analysis are combined with user specification of genuine SQL query. This ensures isolation of affected data and also accuracy with dynamic approach.

The authors further argue that the three-pronged approach endorsed by them are not completely SQLIA proof as there are pros and cons to it. He suggests that developers needs to be made aware of the most updated defensive techniques in order to have an upper hand. They also recommend the researchers make their findings available online as an open source in order to help the community that deal with security.

The authors have presented a finding of their fairly good approach, but with loads of drawbacks as mentioned by the authors themselves. Having said that if each of the three defences are perfected by future developers and researchers, the approach can be highly valuable. One of the problems with the presented defences is that, the authors have talked about certain issues with not escaping wildcard characters, which the hacker could use in password. But in modern defences, password hashing is generated on all websites, which completely takes the wildcard argument off the list.

When explaining about the white list filtering, the authors have not given relevant examples like the usage of Regular Expression (Regex) for the same, which could have been included. Apart from this, the research of the cases is only on the surface level and has not dug deeper.

The authors can further research on second order injection attack significantly for their future work and how their three pronged defensive technique could be used against that.

Though the 3-pronged approach has not been completely adopted in this project, we have tried to emulate some of the techniques, especially the manual defensive coding and testing, the details of which will be disclosed as the chapters progress.

Chapter 3

Project Foundation

The entire project is divided into 3 main parts, i.e, The Company Registration, The Employee Registration and The Login System. All the sections are interconnected. Apart from these, there are other minor subsections like updating of details, account activation, etc. It was crucial that when design of the project was planned, the security of the user data was to be the top priority.

3.1 Company Registration

Company Registration is the process that deals with collection and storage of company information, which is further used to verify their employees who register for our service.

The Figure 3.1 below shows only the basic flow of the entire process. It is only for an understanding of how the company is registered. There are more sophisticated procedures involved, which will be discussed in later chapters. As described in the figure, information from the user is extracted and utilize them to check whether the registration number and the domain of the email provided, is unique in our database. Basically ensuring that the company was not previously registered. If it was so, an error message is sent accordingly. Once the registration is done, email is verified for its validity. If the email is valid, the company account is activated.

The challenge for me was to implement the flow of events shown in Figure 3.1, without compromising the security of the data provided by the user.

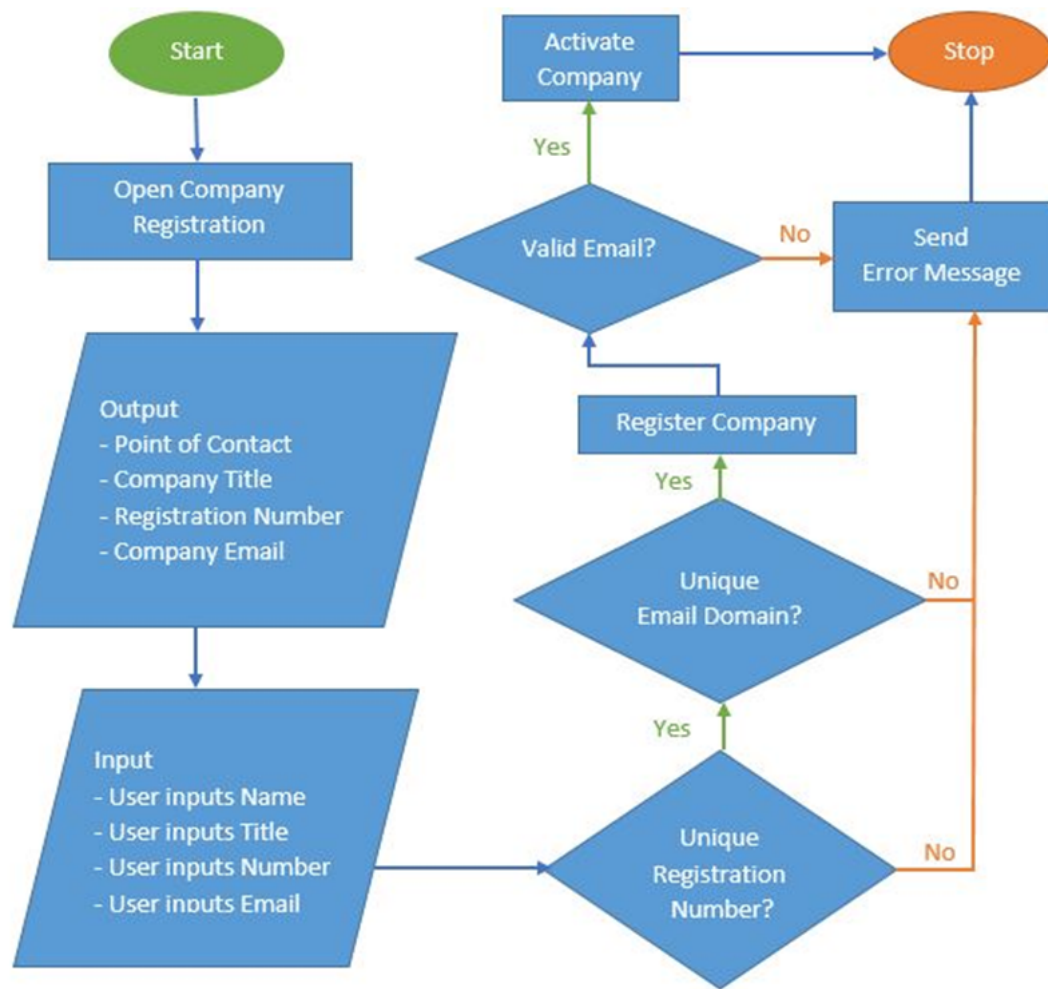


FIGURE 3.1: Company Registration.

3.2 Employee Registration

The procedure for employee registration is not very different from company registration, except that the former has more verification to comply to. In addition to that, mobile attribute is an extra optional feature included. The security apparatus of this process is slightly more complex, which will be covered about, in later chapters.

As demonstrated in Figure 3.2, the user information which is passed on to the database is utilized to check whether the company is registered and active. If so, the function proceeds to check whether the domain of the email provided matches with the company. Once that is taken care, the email is examined whether its unique and has not been previously registered. If all the criteria are met, the employee is registered. The employee has to further verify his email, after which the account is activated. The mobile

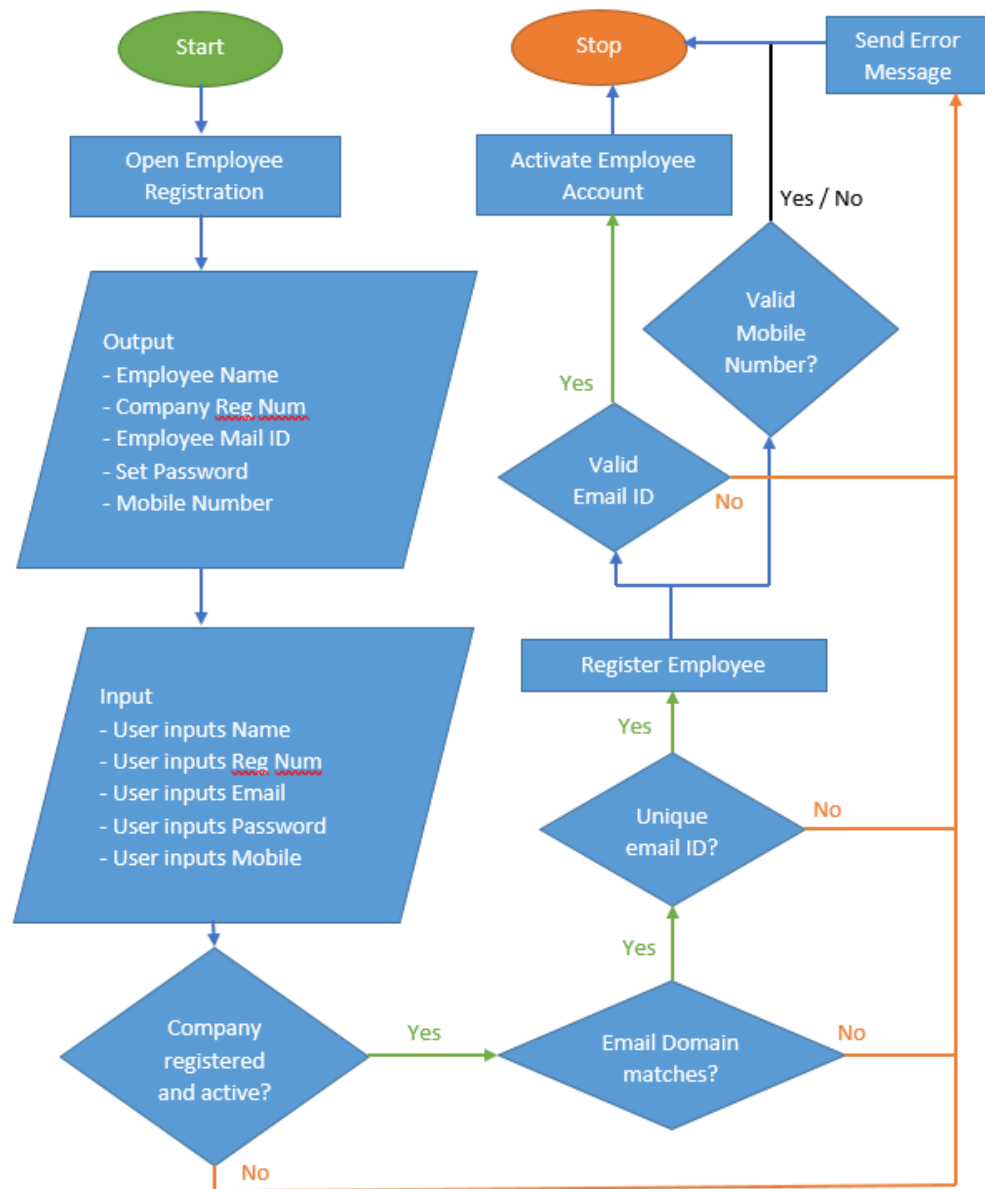


FIGURE 3.2: Employee Registration.

number activation is done alongside the email activation, however, its importance in the employee registration process is not relevant and so is therefore an optional variant.

3.3 Login System

The final section, or the most important section is the Login System, and therefore forms the basis of the project. When the project was initially visualized, only the login

system was planned for. Ironically, major challenges to the project were in the first 2 sections, and therefore the knowledge gained from those were far more.

The login process typically requires 2 main elements – the username and the password. The username here would be the registered employee official email address, and the corresponding password would be the password that was set during employee registration.

The challenge here was to ensure that the right user gets to login. There are several security impediments in place, which have to be satisfied by the user in order to successfully login to the respective company website.

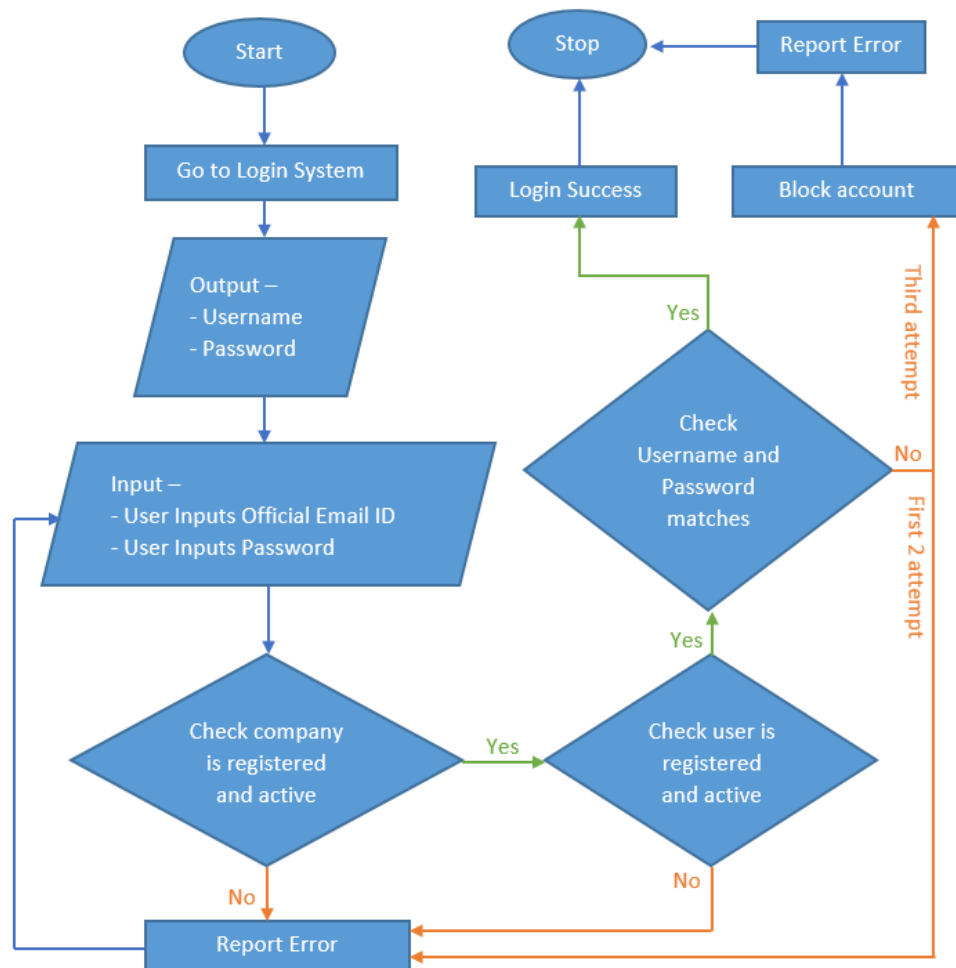


FIGURE 3.3: Login System.

The Figure 3.3 above clearly demonstrates the basic flow of events for the login system. The user inputs are extracted and the information is analyzed to conclude whether the user's company is registered and active. Once that has been determined, the users

email is checked whether it has been registered and is active. The user is then given 3 attempts to successfully login to the respective companys website. If the user fails in the three provided attempt, the account is blocked and no more attempts are awarded. The account has to be re-activated, before the user can attempt to login again. When the user re-activates, he is given 3 further attempts to login and the process continues.

That concludes the fundamental analysis of the project blueprint. The following chapters will concentrate on how the above described problem scenarios were achieved.

Chapter 4

Methodology & Implementation

This chapter deals with the implementation of the database, the web design and the 3 main processes. The methodology and approach to implement the same are discussed as well. The codes and libraries that were crucial to the implementation are also highlighted.

4.1 Database Structure

The basic requirement for any project that depends majorly on back-end database, is to have conceptual model ready. One thing was kept clear from the start that whatever data is stored about the user, it has to have a solid reasoning. There was no point of having extra information which simply fills up the disk space. So unwanted data were excluded from the conceptual level itself. In the end, after a lot of discussion with supervisor, the model looked as shown in Fig. 5.1.

Based on the scenario provided in the Figure 4.1, the following initial assumptions were made:

- Each Company has a unique Registration Number.
- Each Employee has an unique Email ID.
- Each Location has an unique Email and IP Address.
- Each Mobile has a unique Email ID.

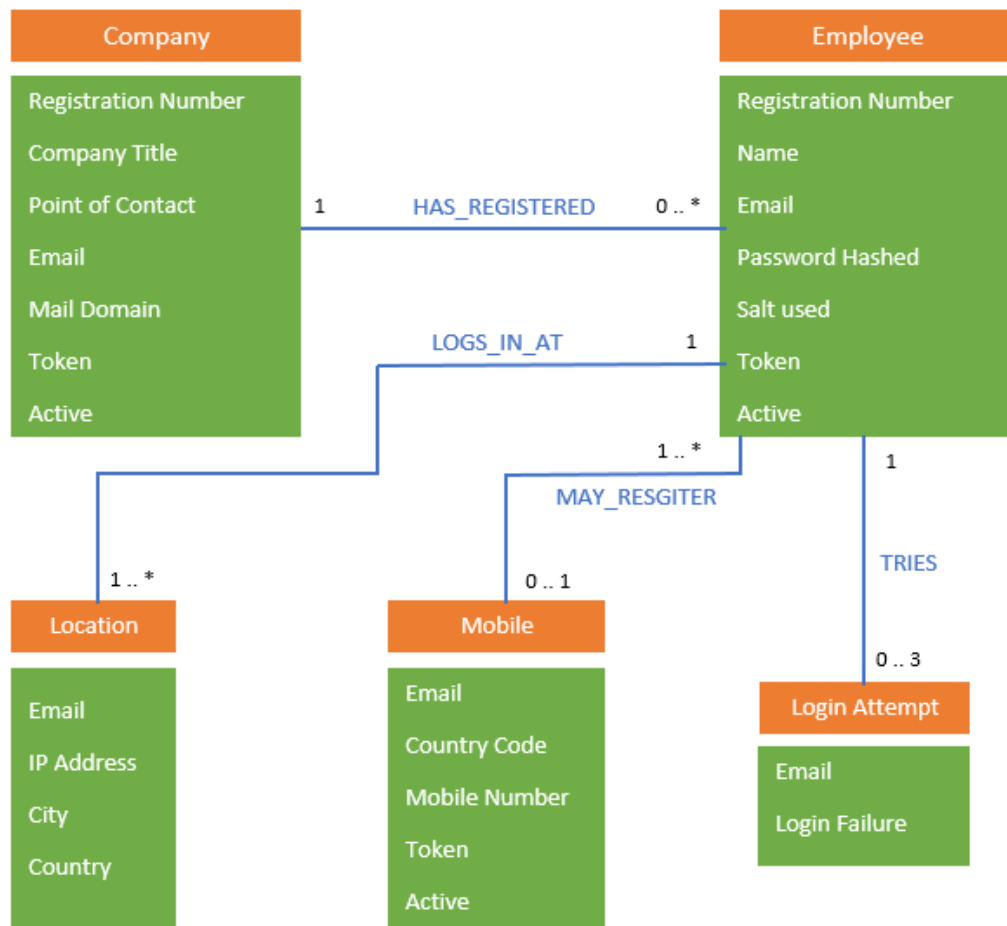


FIGURE 4.1: Conceptual Model: UML Diagram.

- Each Login Attempt also has a unique Email ID.
- A Company can have multiple registered employees
- An Employee can Login from multiple location.
- An Employee can only register Maximum 1 Mobile Number.
- An Employee can attempt to Login 3 times beyond which the account gets blocked.
- An Employee can belong to only 1 company at a time and therefore registered to only 1 account.
- A Location of a particular Email ID and IP Address can belong to only one employee of the respective company.

- A Mobile Number can be registered by multiple persons, though not ideally.
- A Login Attempt attribute can be tracked to a single employee entity.

The above assumptions were established keeping in mind how the project would progress. Once the assumptions were clear, Primary Keys and Foreign Keys were noted for each of the entities. The keys are crucial for creating the database in SQL. Therefore, the following Relational Schema was constructed based on the conceptual model of Figure 4.1:

- Company (**PK - Registration Number**) (Rest of the Attributes - Company Title, Point of Contact, Email, Mail Domain, Token, Active).
- Employee (**PK - Email**) (**FK - Registration Number ref. Company(email)**) (Rest of the Attributes - Name, Password Hashed, Salt, Token, Active).
- Location (**PK - Email, IP Address**) (**FK - Email ref. Employee(email)**) (Rest of the Attributes - Login Attempt).
- Mobile (**PK - Email**) (**FK - Email ref. Employee(email)**) (Rest of the Attributes - Code, Mobile, Token, Active).
- Login Attempt (**PK - Email**) (**FK - Email ref. Employee(email)**) (Rest of the Attributes - Login Failure).

As explained earlier, there was a particular intention behind NOT having any extra attributes in the company and employee tables. Initially, the design had a lot of unwanted attributes such as Date of Birth, Country of Origin, Gender, Employee Address, Home Contact Details, etc. So all those unnecessary elements were trimmed.

Following the relational schema, the SQL commands were implemented in MySQL database and subsequent tables and their contents were created. They were also tested with few hard-coded inputs, as demonstrated in Figure 4.2, with successful results.

In regards to the Active attribute in the company, employee and mobile table, it is used for noting account verification. The code is accordingly changed in the active attribute as and when the details are verified, enabling the users to utilize our service. The Token attribute is also used for verification purpose. The Mail Domain attribute was initially

```

+-----+-----+-----+-----+-----+-----+-----+-----+
| regNum | title | pointContact | email | mailDomain | token | active |
+-----+-----+-----+-----+-----+-----+-----+
| 123 | BRAM Films | Arjun KP | ak750@kent.ac.uk | kent.ac.uk | 648859894740571 | 1 |
| 543543435 | Dhritraksh | Mithun K P | mits@driks.com | driks.com | 987979340402269 | 1 |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> select * from employee;
+-----+-----+-----+-----+-----+-----+-----+-----+
| regNum | name | email | hash | salt | token | active |
+-----+-----+-----+-----+-----+-----+-----+
| 123 | Arjun KP | ak750@kent.ac.uk | 9940df486922c65c53fec3f6192997336067e8fa5c96e | be000a352f0dc316306084a93 | 900602339870093 | 1 |
| 543543435 | Mithun K P | mits@driks.com | lafcdf975f6caf90b46ac8c93012467ff8271dcfdbf46 | c8398d6841539b7a20ac2eaaad | 808416301186446 | 1 |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> select * from location;
+-----+-----+-----+-----+-----+-----+-----+-----+
| email | IPAddress | city | country |
+-----+-----+-----+-----+-----+-----+-----+
| ak750@kent.ac.uk | 129.12.225.235 | Canterbury | United Kingdom |
| mits@driks.com | 129.12.216.248 | Canterbury | United Kingdom |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> select * from login;
+-----+-----+-----+-----+-----+-----+-----+-----+
| email | notlogged |
+-----+-----+-----+-----+-----+-----+-----+
| ak750@kent.ac.uk | 0 |
+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> select * from mobile;
+-----+-----+-----+-----+-----+-----+-----+-----+
| email | code | mobile | token | active |
+-----+-----+-----+-----+-----+-----+-----+
| ak750@kent.ac.uk | 44 | 7752511531 | 9197931 | 1 |
| mits@driks.com | 44 | 7752511531 | 2803772 | 1 |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

```

FIGURE 4.2: SQL database

not planned, but it had become significant as the project progressed, which will be discussed in later chapters. The salt and hashed Password were attributes included based on the security requirements, which also are discussed in further chapters.

4.2 Website Logo & Design

With the database elements in place, the focus turned towards the design of the website. Before the website design could be initialized, the name of the website and the logo design was to be completed. Since, the concept of this project is about securely connecting employees to their company website, the idea was to have a name that revolved around the word "Connect". So the website was named "Dot.Connect". To complete the logo, 4 large dots connecting in the shape of a square was fixed to the title as shown in Figure 4.3.

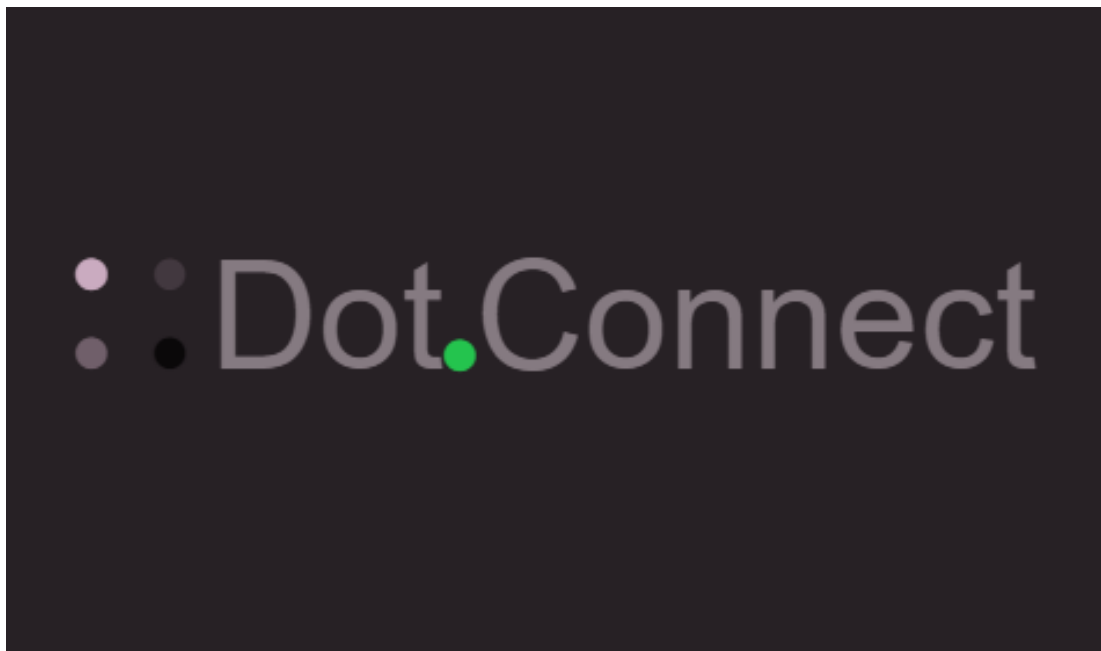


FIGURE 4.3: Dot.Connect Logo Design

Once the logo shape was fixed, the attention progressed to color. To fix the colors, the website background color had to be confirmed because the logo needed to stand out against the background color. If the background was dark, the foreground elements had to be light and vice-versa. Contrast is the basic rule of design as explained in [6].

Before arriving at a decision on the website design, there were lot of trial and error around different types of webpages, by attempting to design the website initially based

on what's shown in [7], but the texture was difficult to implement. Also, had a try on the design demonstrated in [8], but it was more suited for mobile app than for a website, so had to drop the idea.

The choice was finally firmed on [9]. Design and color was implemented as demonstrated in [9]. It took time and effort to achieve the strikingly similar elements, but was able to execute it.

The first webpage designed for the entire website was the login page. So, needed careful study on the composition of the elements, which will be similar on rest of the webpages. This was important to maintain because it ensured aesthetics of the website. There was a lot of tweaking initially, but was able to fix the mind to one particular composition, which could fit for all the pages in the website. The application of the design looked as shown in Figure 4.3 and Figure 4.4, and the composition of elements were reflecting as in Figure 4.5.

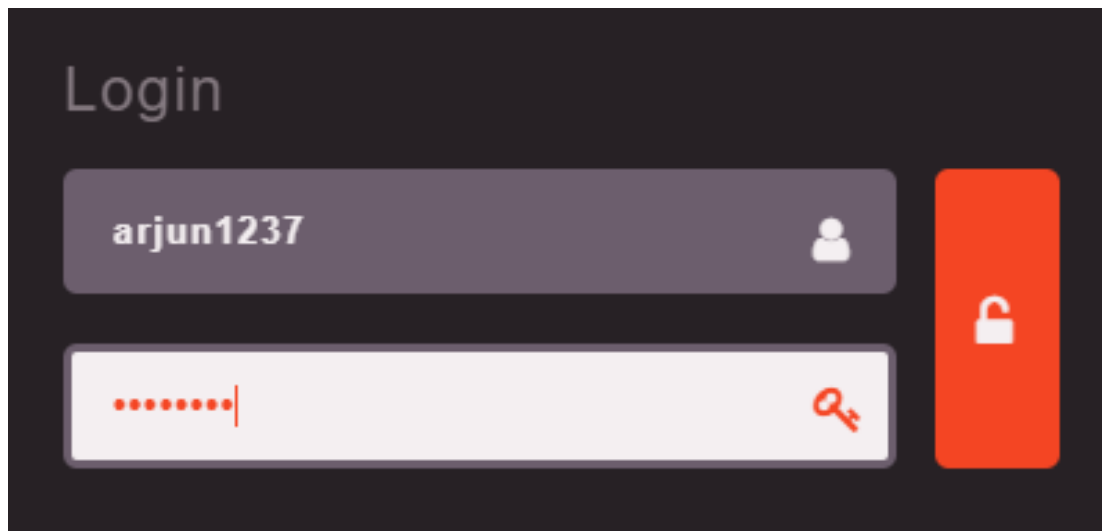


FIGURE 4.4: Login System element design

4.3 Implementation of Web-Design

To implement the design as shown in Figure 4.5, the styling had to be right. In previous web service project[10] that was attempted, there were a lot of issues with positioning of the elements in different systems with non-identical display resolution. So there was a need to position all the elements to a fixed boundary in a given space, which is not going to change with change in display resolution.



FIGURE 4.5: Website Design & Composition

The following CSS code was important in affixing the layout where all my elements will be concentrated on:

```
.layout1{
    position: absolute;
    padding: 10px 25px;
    left:50%;
    top:50%;
    transform: translateX(-50%) translateY(-50%);
    width: 1100px;
}
```

So, the above code took care of the display resolution issue. With that sorted, there was also a need to create a division within the layout between the logo and the part where authentication service is implemented. As shown in Figure 4.5, a vertical line was drawn between them to separate the two with the help of following CSS codes:

```
.vertical-line{
    width: 1px;
    background-color: black;
    height: 100%;
    float: left;
    position: absolute;
    left: 500px;
}
```

When an input was in focus, there was a need for change in color scheme. With the help of JQuery, on-focus and on-blur features of CSS were altered on input elements, like the one in Figure 4.1. The following jQuery codes is a sneak peek into focus and blur changes in attribute on one of the input element:

```
user.focus(function(){
    user.css({borderColor: "#6c5e6d"});
    user.attr("placeholder", "");
    user.css({color: '#f44523', backgroundColor: '#f4eff1'});
```

```
        userIcon.css({color: '#f44523'});
    })

    user.blur(function(){
        user.attr("placeholder", "Official Email ID");
        user.css({color: '#f4eff1', backgroundColor: '#6c5e6d'});
        userIcon.css({color: '#f4eff1'});
        warning.html("");
    })
```

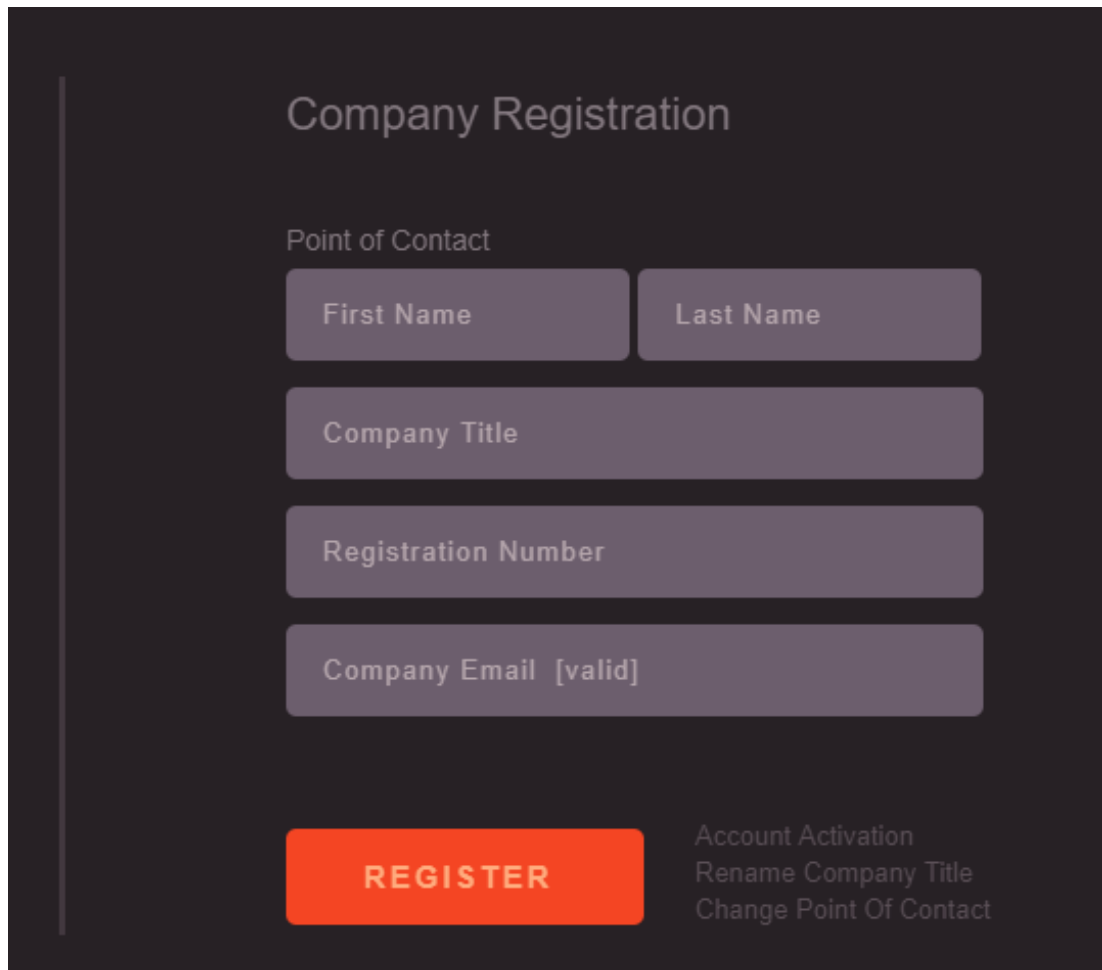
Similar codes were applied on all the input throughout the website.

4.4 Company Registration

For the company to be registered, 4 attributes are extracted from the user – Company Title, Registration Number, Point of Contact and his Email ID as in Figure 4.6. The other 3 attributes are gathered automatically from calculations.

Once the user has provided with input, on clicking Register button, several checks are conducted. The first check is whether all the necessary input has been provided by the user. Following that, all the entries are validated. If the user did not pass the checks on his input, an alert message would pop up, restricting him from registering his company as in Figure 4.7. It is one way to block user from wrong entries.

Following the client side validation, the results are passed on to server side, where again inspection is carried out. On the server side, there are 3 essential checks on all inputs - checking whether the necessary inputs are filled, sanitizing them with *stripslashes()* and *htmlspecialchars()* functions and then a final validation check using Regex. The features of web security are discussed in detail in [11], which will be touched upon in later chapters. These are some of the basic checks on any inputs and is part of defensive coding technique that was implemented. If the programmer forgets to put these checks and the user is able to find the loophole, he can take advantage of it, which can lead to data stealing, data manipulation and worse - erasing of the entire data.



The image shows a 'Company Registration' form on a dark background. The form is titled 'Company Registration' in a large, light-colored font. Below the title, there is a section labeled 'Point of Contact'. This section contains four input fields: 'First Name', 'Last Name', 'Company Title', and 'Registration Number'. Below these fields is a single input field for 'Company Email [valid]'. At the bottom of the form, there is a prominent orange button labeled 'REGISTER'. To the right of the 'REGISTER' button, there is a list of links: 'Account Activation', 'Rename Company Title', and 'Change Point Of Contact'.

FIGURE 4.6: Company Registration attributes

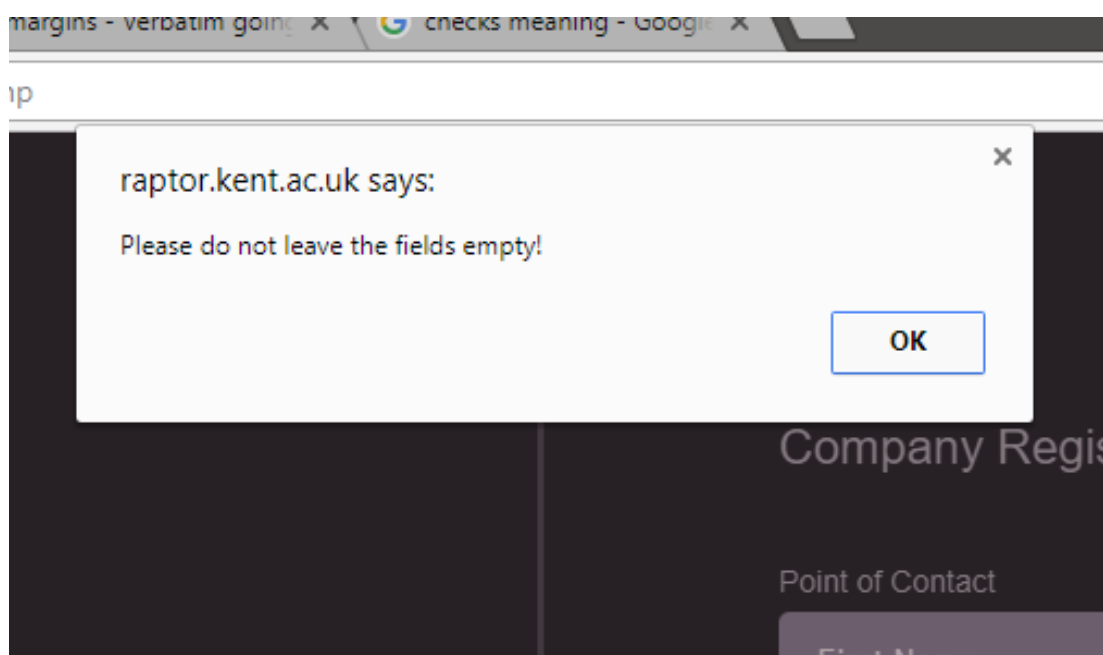


FIGURE 4.7: Alert Message

With input validation done, verification is carried out by first checking whether there exists company registered with similar registration number or emails with similar domain. This will confirm that the company registering for our service are not previously registered, avoiding duplicate entries. This is one of the reason why a mail domain attribute was required in the company table of the database. Since, the mail domain is now present as an attribute, duplicate entries can be prevented as the mail domain are unique to the company they represent.

Following that, a cryptographically secure random number is generated to create a token. The reason token attribute was required in company registration process is to verify the email address of the point of contact. Initially, the user details are registered onto the database along with token details, but without activating it, i.e., having the active attribute set to 0 as in Figure 4.8.

```
mysql> select * from companyInfo;
```

regNum	title	pointContact	email	mailDomain	token	active
123	BRAM Films	Arjun KP	ak750@kent.ac.uk	kent.ac.uk	648859894740571	0

FIGURE 4.8: Company BRAM Films with active attribute set to 0

A mail is sent to the user with generated token number attached to the URL, which when clicked, the token is compared with the one stored in database and the account turns to active mode by updating the active attribute to 1 as shown in Figure 4.9.

```
mysql> select * from companyInfo;
```

regNum	title	pointContact	email	mailDomain	token	active
123	BRAM Films	Arjun KP	ak750@kent.ac.uk	kent.ac.uk	648859894740571	1

FIGURE 4.9: Company BRAM Films with active attribute set to 1

The activation mail is sent through PHPMailer. The use of *mail()* function in PHP was not a bad idea either, but suppose in future, there is a requirement in the service for sending a HTML-heavy mail across, *mail()* function would not support any HTML elements, whereas PHPMailer does.

Also, the *mail()* function usually sends via a local mail server, typically fronted by a sendmail binary on Linux, BSD and OS X platforms, however, Windows usually doesn't include a local mail server; PHPMailer's integrated SMTP implementation allows email sending on Windows platforms without a local mail server.^[12]

For unexplained reason, if the activation email was not received, a link is sent for activating the account when provided with the email ID of the Point of Contact. In case, there was a mistake in Contact Person Name or the Company Title details, there are features in place for amending them as well.

Suppose at any stage, there is a problem with the registration process, an error message is promptly displayed to the user accordingly. The error message is handled from a single file with the name *errorMsg.php* and is common for any error request from any webpage within the "Dot.Connect" website.

If the process is successful, the company is registered and active. This enables the employees of the company to register with our service, which is next phase of project.

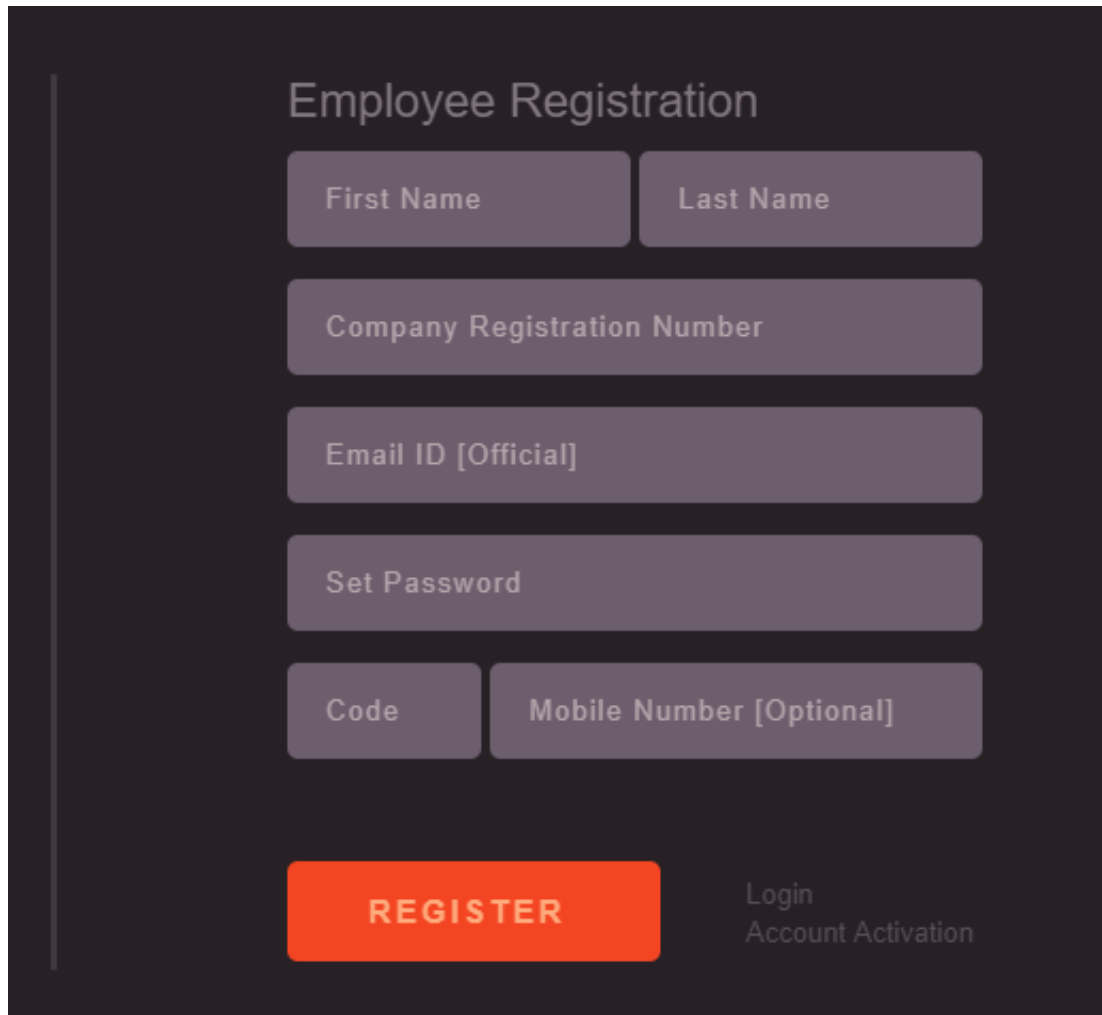
4.5 Employee Registration

In the employee registration process, any employee of the respective registered company, will be able to register his account with us. To enable the registration, there are certain essential requirements - Name (as usual), Email, Company Registration Number, Password to be set, Mobile Number (optional element) as shown in Figure 4.10. The password is not directly stored in the database due to security reasons. The rest of the attribute, which are calculated and directly included in the database are salt, hash, active and token.

As in the previous process, similar client-side validation is done to ensure the user is entering all the essential inputs. It is further ensured that the inputs are genuine.

With client side validation done, the user inputs are passed onto the server side, where again validation takes place as mentioned in the previous section. Once all the validation checks are passed, the actual process of the registration is initiated.

The first verification is to check if the company is registered and active. With the registration number provided by the user, it is verified if it exists in the database and then check if the active attribute is set to 1. Once that is taken care, email domain is verified from the email ID provided, if it matches with the respective company. Following that, the email is also checked if its not been registered previously to avoid duplication of entries.



The image shows a dark-themed web form titled "Employee Registration". It contains several input fields: "First Name", "Last Name", "Company Registration Number", "Email ID [Official]", "Set Password", "Code", and "Mobile Number [Optional]". At the bottom, there is a prominent orange "REGISTER" button and a link for "Login Account Activation".

FIGURE 4.10: Employee Registration attributes

After the verification, a salt is created. The salt and the password are then mixed and a hash is generated using the mix. The process of creating one is explained in chapters to follow. A token is also generated as part of the process for email verification later on. With all assessment ticked right, the employee account is registered. The password, as explained earlier, is not stored in the database. The details of which are shown in Figure 4.11.

```
mysql> select * from employee;
```

regNum	name	email	hash	salt	token	active
123	Arjun KP	ak750@kent.ac.uk	9940df486922c65c53fec3f6192997336067e8fa5c96e	be000a352f0dc316306084a93	900602339870093	0

FIGURE 4.11: Employee Detail storage in database

An activation mail is then sent to the email associated with the account with a token attached to URL, which when clicked activates the account by setting the active attribute to 1 from 0.

The IP Address location is also stored in a separate table as shown in Figure 4.12. This information is important to determine if the user is logging in from same or different location. The security reasons for the same is explained in sections to follow.

```
mysql> select * from location;
```

email	IPAddress	city	country
ak750@kent.ac.uk	129.12.225.235	Canterbury	United Kingdom

FIGURE 4.12: Location Table detail storage in database

If the mobile number details were provided by the user, which was an optional element, the numbers also goes through validation check. A token is created for mobile verification later on. Mobile details are stored accordingly in the database as displayed in Figure 4.13. An SMS is then sent to the provided mobile number for verification as shown in Figure 4.14. Once the user provides the right token, the mobile number gets activated.

```
mysql> select * from mobile;
```

email	code	mobile	token	active
ak750@kent.ac.uk	44	7752511531	9197931	1

FIGURE 4.13: Mobile Table detail storage in database

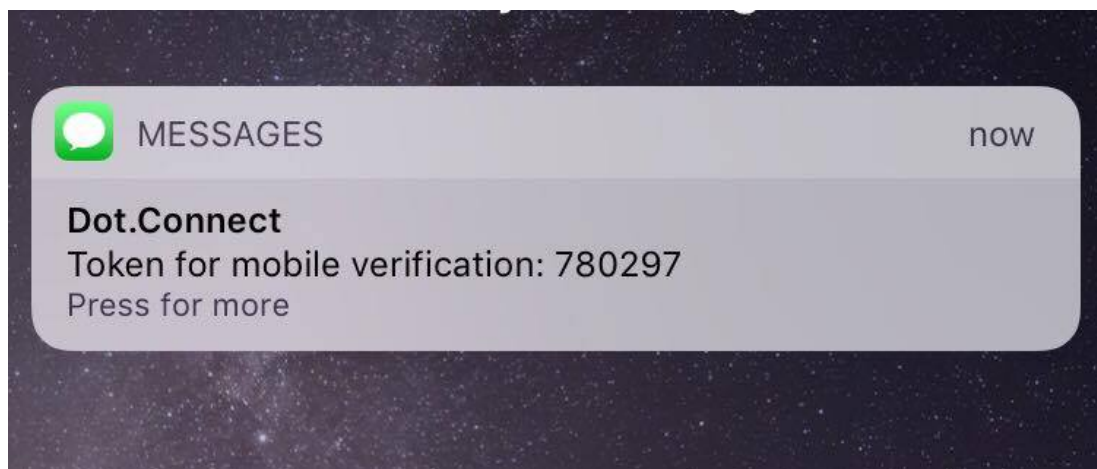


FIGURE 4.14: Token for mobile verification through SMS

The mobile number as explained in earlier chapters, is not required for the registration. But, in future, if mobile number is used for resetting password, it can become a crucial element.

The SMS is sent through the use of Text Local service, which was registered to. Basically, with the account name and password in place, few other details are passed on through POST method by using a curl functionality[13] in the PHP as shown below:

```
$vars = 'uname='.$user.'&pwd='.$pwd.'&message='
        . $msg.'&from='.$from.'&selectednums='.$number.'&info=1&test=0';
$curl = curl_init('http://www.txtlocal.com/sendsmspost.php');
curl_setopt($curl, CURLOPT_POST, true);
curl_setopt($curl, CURLOPT_POSTFIELDS, $vars);
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
echo $result = curl_exec($curl);
curl_close($curl);
```

Initially, the biggest challenge was to implement the email and SMS verification. But, it turned out to be the least challenging with the PHPMailer library easily available and the Text Local SMS service and the curl function doing the rest of the part. The SMS service was not available for certain countries like for users from US and Canada as their governments have blocked the SMS Alerts from being sent to North Americans from outside the countries for security purpose. There was a way out for this. That was by having a separate number for North America and that meant extra bucks from the pocket, which was avoided.

With success in registration of the employee, the user can proceed to use our authentication system to login to her/his respective company website. Suppose, the activation was not successful for some reason or the employee account was blocked, it can be activated again by using the appropriate link in webpage and providing the required information.

4.6 Authentication or Login System

The final phase is the Login System. The process requires only 2 inputs from the user and that is the username, which is the official email ID registered with us and the password that was provided to us as shown in the Figure 4.15.

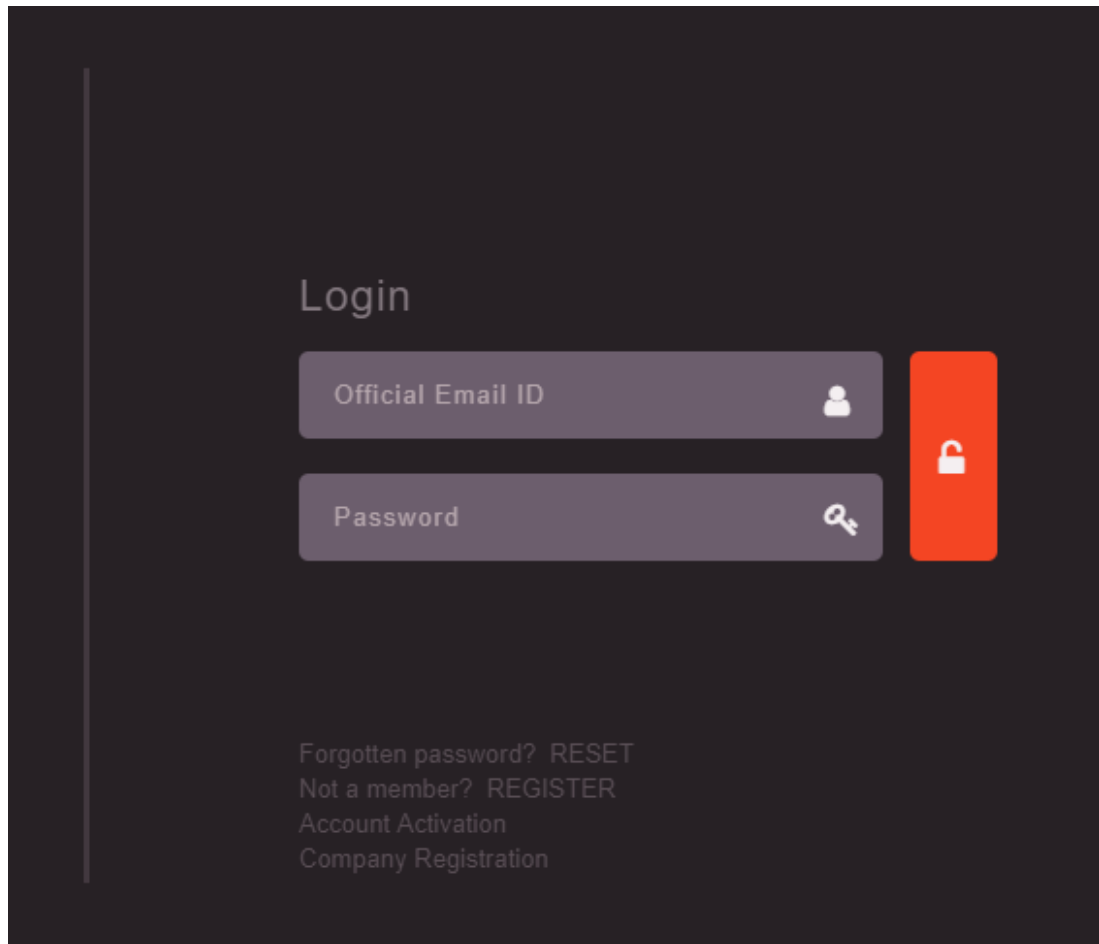


FIGURE 4.15: Login System attributes

The client side and server side validation that follows the user input, is similar to the previous 2 processes. With validation checks complete, the domain from the email provided, is extracted. With domain information in place, the company corresponding to the domain could be checked if it exists and is active. Then, the email provided could be checked if it exists as well, in the database and is active too. If it turns out active, all the employee information are extracted from employee table which includes salt and hash data.

The location information of the user from the server is then extracted. By comparing the current login location with the historical location data, IP address is verified by matching with any one of them, else a warning mail is sent saying that there is a login attempt from different location. However, that does not affect the login attempt in any way and so the process resumes normally. The password provided by the user and stored salt extracted from the database are combined and hashed. The hashed value is

then compared with the corresponding hash stored in the database. If both the values match, the login is successful and the page proceeds to the respective company website.

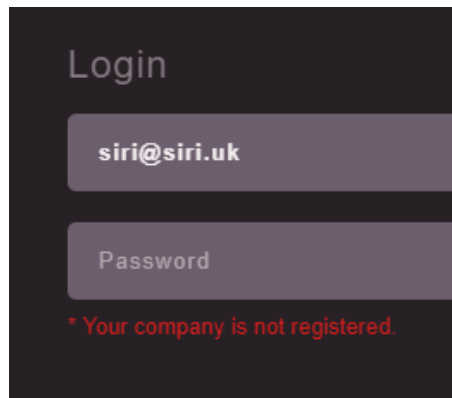


FIGURE 4.16: Error Message for company not registered

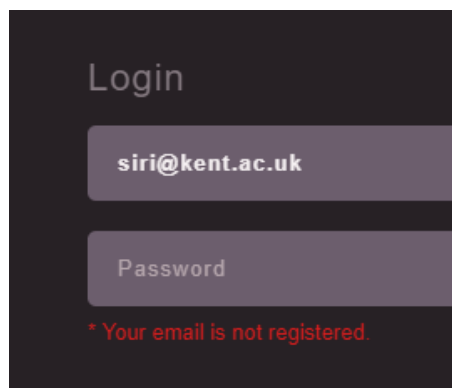


FIGURE 4.17: Error Message for employee account not registered

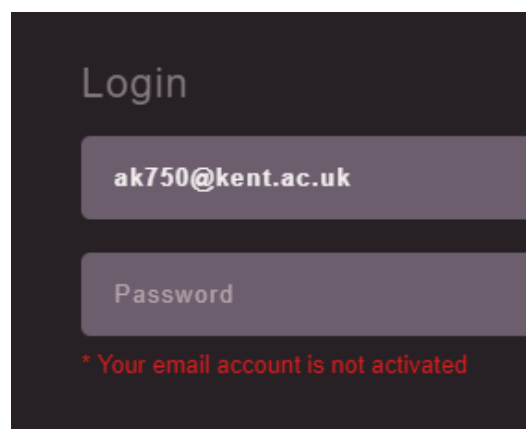


FIGURE 4.18: Error Message for employee account not being active

Suppose, if the user email is verified, but his credentials does not match, he is given 2 more attempts to login. If the user fails to do so, the account gets blocked. The account then has to be re-activated in order to login.

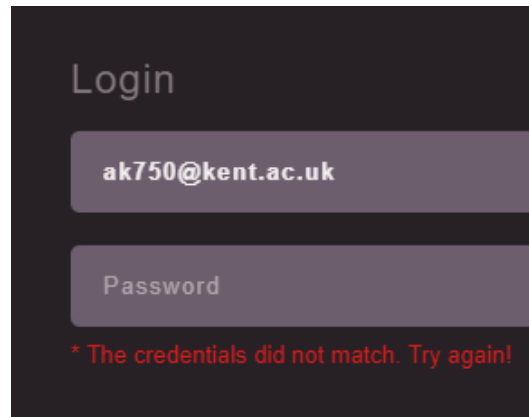


FIGURE 4.19: Error Message for not matching credentials

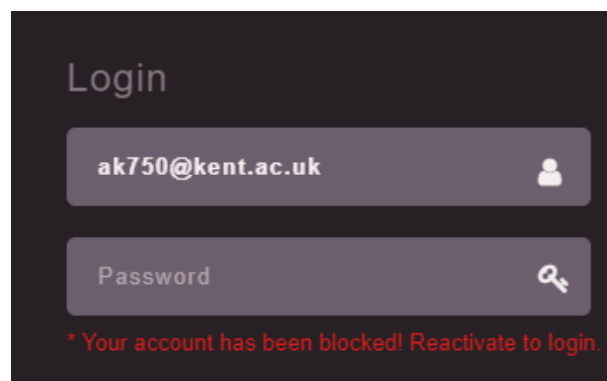


FIGURE 4.20: Error Message for account block

Ajax was used for this phase, unlike the other 2 previous phases. This ensures that in case of login attempt failure, the error messages were appearing in the same page itself. Only the error message was updating, instead of redirection to a separate page. This ensured smoothness as demonstrated in Figure 4.16, 4.17, 4.18, 4.19 and 4.20.

The Ajax request was sent and received with the help of following codes:

```
var http = new XMLHttpRequest();
var url = "loginCheck.php";
var params = "user="+user.val()+"&pass="+pwd.val();
http.open("POST", url, true);

//Send the proper header information along with the request
http.setRequestHeader("Content-type", "application/x-www-form-urlencoded");

http.onload = function () {
```

```
// get response from loginCheck.php
warning.html(this.responseText);

// if credentials are right
if(warning.html().endsWith('success!')){
    warning.html('');
    val = true;
    submit.trigger( "click" );
}

// if credentials are NOT right
else{
    unlock.css({visibility: "visible"});
    spin.css({visibility: "hidden"});
    pwd.val('');
}
};
http.send(params);
```

With this the implementation part comes to an end. The following chapter will deal with the security procedures in place and also how they help avoid the most popular hacks.

Chapter 5

Precautionary & Security Measures

The obligation to protect valuable data is as old as history. As far as cyber security is concerned, the global cost of cybercrime is greater than the combined effect on the global economy of trafficking in marijuana, heroine and cocaine.[\[14\]](#) In this chapter, major website vulnerabilities of modern times and the security measures put in place to ward off threats are uncovered, along with user input restrictions used to protect the website from users making witless mistakes.

5.1 Web Vulnerabilities

During the development of the Dot.Connect website, it had to be noted that a lot of companies will be catered to, and therefore a lot of data for potential hackers to exploit. As many checks as possible had to be implemented to make it difficult for the hackers to have any headway. So, there was a need to have an idea of the kind of vulnerabilities, both from client end and server end.

5.1.1 Cross Site Scripting (XSS)

XSS takes advantage of the vulnerabilities on webpages where the page or the site unveils information from the unchecked data provided by the user. XSS enables attackers

to inject client-side scripts into web pages viewed by other users. It may be used by attackers to bypass access controls such as the same-origin policy. Bug bounty company HackerOne in 2017 reported that XSS is still a major threat vector. XSS effects vary in range from petty nuisance to significant security risk, depending on the sensitivity of the data handled by the vulnerable site and the nature of any security mitigation implemented by the site's owner.[15]

Essentially, in order for cross site scripting to work, the website must be vulnerable to script injection and there must be some kind of an interaction between the users. Consider a website where it allows people to send unsanitized messages to a particular user, like in Figure 5.1 with a message: "This is a test". When the user, to whom it has been sent to, reads them, the message would appear normal like in Figure 5.2.



FIGURE 5.1: Input: Normal Message

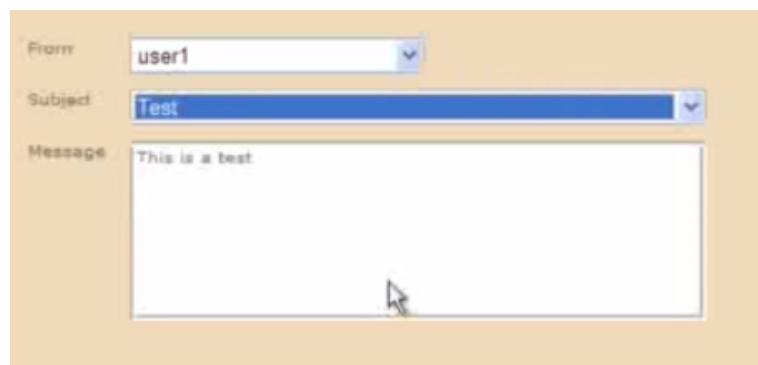
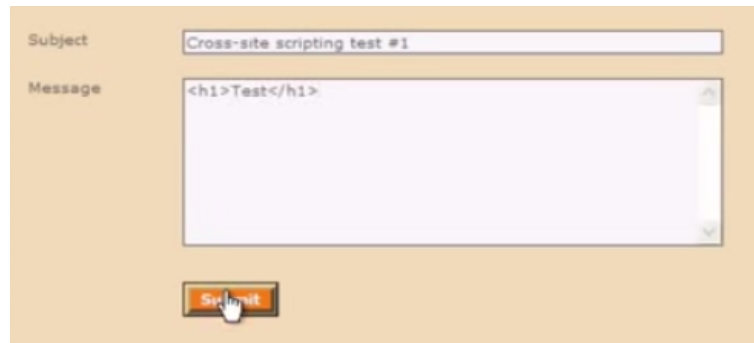


FIGURE 5.2: Output: Normal Message

Suppose, the hacker sends a message with some HTML tags in it as in Figure 5.3, the message would appear to the user with bold and bigger font as shown in Figure 5.4. Hence, a first indication that the inputs have not been sanitized. If the hacker finds this out, he could try something more malicious like including JavaScript within script tags, as in Figure 5.5.



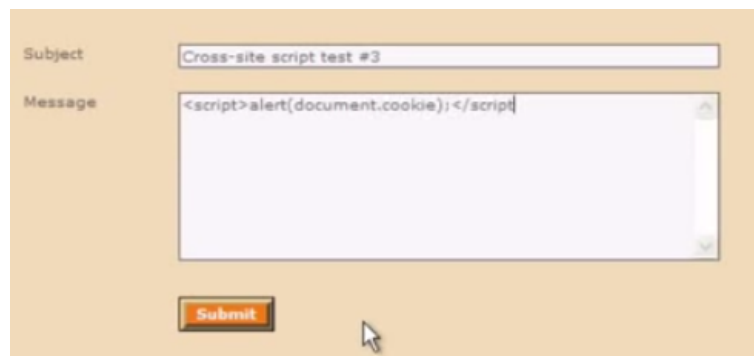
A web form with a light orange background. It has two input fields: 'Subject' and 'Message'. The 'Subject' field contains the text 'Cross-site scripting test #1'. The 'Message' field contains the HTML code '<h1>Test</h1>'. Below the 'Message' field is a 'Submit' button with a mouse cursor hovering over it.

FIGURE 5.3: Input: Message with HTML tags



The same web form as in Figure 5.3, but now the 'Message' field displays the rendered HTML as 'Test' in a large, bold, black font. A mouse cursor is hovering over the text 'Test'. The 'Subject' field still contains 'Cross-site scripting test #1' and the 'From' field now shows 'user1'.

FIGURE 5.4: Output: Message with HTML tags



A web form with a light orange background. It has two input fields: 'Subject' and 'Message'. The 'Subject' field contains the text 'Cross-site script test #3'. The 'Message' field contains the JavaScript code '<script>alert(document.cookie);</script>'. Below the 'Message' field is a 'Submit' button with a mouse cursor hovering over it.

FIGURE 5.5: Input: Message with some malicious code

So when the user clicks on the message, he gets an alert message with cookie details displayed as in Figure 5.6.

This is just a sneak peek into what XSS can accomplish. The more deep XSS goes, the more damage it can do. There are 2 type of XSS attacks: Persistent and non-Persistent. The former ones are more dangerous as it has the capability to spread from one to another. The ways to avoid these attacks are discussed in sections further.



FIGURE 5.6: Output: Message with some malicious code leading to alert

5.1.2 SQL Injection

SQL injection is a code injection technique, used to attack data-driven applications, in which nefarious SQL statements are inserted into an entry field for execution. SQL injection must exploit a security vulnerability in an application's software, for example, when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed.^[16]

There are many levels of attacks, but if the website does not have basic protection in place, even a basic SQL injection can easily manipulate or extract database information. Consider a website that is susceptible to SQL injection. The site has no check for the right characters being used and it has 2 inputs from the user, namely username and password. Suppose, an SQL query passed in PHP is:

```
// lets assume user and pwd are 2 attributes...
// ...in the userDetails table of the database.
// lets assume $username and $password are....
// ...php variables for the user inputs
$sql = "SELECT * FROM userDetails WHERE
        user = $username and pwd = $password";
```

Consider a scenario where input was 'jsmith' for username and 'password123' for password, the SQL query would reflect the following:

```
SELECT * FROM userDetails WHERE user = 'jsmith' and pwd = 'password123';
```

But that's not what the potential hackers are going to try. Suppose, the hacker knows the username of a potential target. Lets assume the target was 'jsmith'. The hacker inserts the username as:

```
jsmith';--
```

And the password as:

```
1234 (remember the password correctness does not matter here)
```

The command would look like this when sent to SQL:

```
SELECT * FROM userDetails WHERE user = 'jsmith';-- ' and pwd = '1234';
```

However, SQL would read the above command as:

```
SELECT * FROM userDetails WHERE user = 'jsmith';
```

This is because the single-quote (') just after jsmith and a semi colon (;) that follows would end the sentence and the double dash (--) that follows the semi colon would comment out rest of the query in SQL. Hence, the single quote after the double dash and rest of the commands (' and pwd = '1234';) does not matter. So, the above query would run, without the final condition. In practice, that is a huge security breach as the above vulnerability could have lead to the users personal account.[\[17\]](#)

Suppose the hacker typed further in username input as:

```
jsmith';DROP TABLE userDetails;--
```

The above query that follows jsmith would completely delete the table. The biggest danger is, if the website is vulnerable easily, you do need even need to know the username, for example, if the potential hacker inserts the following as username:

```
' OR 1=1;--
```

From the above input, the query will be read as:

```
SELECT * FROM userDetails WHERE user = '' OR 1=1;-- '
```

This means, the `user = ''` is incorrect but there is an "OR" added followed by a condition `1=1` which is always true. So, since any one condition has to be true and `1=1` is true, therefore the query is passed without any error, thereby logging into the user account. Usually, the account of first user listed in the database is compromised in such cases.

5.1.3 Brute Force and Dictionary attack

A brute force attack is an online offensive where the hacker tries to ingress into an account by finding the right username and password combination. Its a trial and error method, almost always executed by an automated software.[18] Brute Force can become cumbersome at times, without yielding any result. It also depends on the speed of the computer and how powerful it is.

A lot of hackers get access to databases directly through advanced hacking techniques. So, if the passwords are stored directly, then you are in trouble. Modern technique is to have hashes of passwords stored instead of having passwords itself.

When hackers target a particular hash by brute force, they try from words of fixed size, say 7 words. The algorithm start hashing with AAAAAAA, and then move on to AAAAAAB, and so on for different character sets. They compare the hashes one by one and see if it matches. This technique can be useful if the passwords are not very long and is either only in lowercase-case or uppercase-case.[18] In Dot.Connect website, blocks for such techniques were created, which are discussed in further sections of the paper.

In cryptanalysis and computer security, a dictionary attack is a technique for defeating a cipher or authentication mechanism by trying to determine its decryption key or passphrase by trying hundreds or sometimes millions of likely possibilities, such as words in a dictionary or previously used passwords[19]. There are many software available for such purpose like the popular one, John The Ripper[20]. When directed to vulnerable website, the software tries each and every passwords stored, one by one, and checks whether it is able to login. It keeps trying until it succeeds. This method of hacking is still popular as people usually set passwords they can easily remember and

on most occasions reusing their old passwords which may have been in list of passwords the hackers have acquired. Therefore allowing hackers to easily crack. There are ways to create barriers for these and is discussed in further sections.

5.2 Measures Taken

The web breaches are so advanced that there are software like The Mole, that does all the work for you by extracting information, when provided with vulnerable URL[21]. Therefore, precautions, enabled by strong security, are necessary. For Dot.Connect website, the kind of barriers that can be put in place was noted beforehand in order to stop those attacks.

5.2.1 REGEX Validation and other checks

The first barrier for any hacking on a website, is to monitor the user input from the client-end itself. Most hackers find vulnerabilities by posing as a website user and analyzing each and every potential unguarded user input points.

Therefore, in Dot.Connect website, on all the pages that was designed, there were checks and conditions for each and every user inputs. Inputs where there was specific requirement needed for allowing only numerical values, the input type was given as 'number' in the form itself, under HTML, which ensures the user cannot type anything other than numbers. For Example, like the following one:

```
<input type="number" name="compRegNumber"
      placeholder="Registration Number" />
```

Another way to restrict the user from exploring unwanted characters is to have Regular-Expression(Regex) validation in place. The Regex makes sure the user inserts what is required of him by ensuring inputs did not fall out of boundary of acceptable characters. The following codes were some of the Regex used from the client end for email types and name types respectively:

```
// consider variable a -> email
```

```
function checkmail(){
    if((/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/).test(a))
        return true;
    return false;
}

// consider variable name - > any name
function checkName(name){
    if(/^[A-Za-z0-9]+(?:[ _-][A-Za-z0-9]+)*$/).test(name))
        return true;
    return false;
}
```

The above Regex ensures the attacks are not carried out from the user inputs. Similar Regex was also used for validating when the inputs are transferred to server-side, like the following function:

```
function validate($input, $type, $msg = 'Input'){
    if($type == 'name'){
        if (!preg_match("/^[a-z0-9_\-\s]+$/i",$input)){
            header("Location: errorMsg.php?msg=$msg&er=1");
            die();
        }
    }
    else if($type == 'num'){
        if(!is_numeric($input)){
            header("Location: errorMsg.php?msg=$msg&er=1");
            die();
        }
    }
    else if($type == 'email'){
        if (!filter_var($input, FILTER_VALIDATE_EMAIL)) {
            header("Location: errorMsg.php?msg=$msg&er=1");
            die();
        }
    }
}
```

```
    }
}
else if($type == 'empty'){
    if (empty($input)) {
        header("Location: errorMsg.php?er=1");
        die();
    }
}
}
```

In this way, same inputs go through multiple Regex checks so hackers don't find any loopholes in the system. If the input fails the Regex Validation, error message is displayed accordingly.

5.2.2 Sanitization

Once the inputs are passed to server-side, there was a need for more barriers to be placed. For every input that is transferred either through POST or through GET method in Dot.Connect website, it has to go through sanitization process. The following codes were instrumental in ensuring the same:

```
function normalize($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
```

The above mentioned 3 functions are in-built in PHP. **trim()** function is not a security barrier. It just trims out the unwanted spaces at both ends of the inputs and also removes new line if it exists at both ends of the string.

The **stripslashes()** function removes all the back-slashes from the given string[22]. So, **echo stripslashes("Is your name O'reilly?");** would print as

Is your name O'reilly?

The **htmlspecialchars()** function recognizes special characters in a string and converts them to HTML entities. Assume a user inserts the following code into as an input, which redirects the victim to a particular site:

```
<script>location.href('http://www.hacked.com')</script>
```

- If the above code is sanitized with *htmlspecialchars()*, the string would appear like the following:

```
&lt;script&gt;location.href('http://www.hacked.com')&lt;/script&gt;
```

The code was now safe to be displayed on a page and is a good barrier for XSS attacks.^[23] The above functions also helped curb any SQL injections.

5.2.3 Prepared Statements

For Dot.Connect website, SQL commands are executed through the PHP functions. PHP currently has 2 best practices for safe SQL query execution, and both of them are good enough. Those are MySQLi and PDO. Both support prepared statements, which forms the final barrier for SQL injection. The advantage PDO has over MySQLi is that the former is better option if the database is moved from one RDMS to another, because no changes need to be made. MySQLi feels procedural, whereas PDO is truly Object Oriented and therefore feels more intuitive.

The following code demonstrates a basic SQL query with prepared statements:

```
function getPerson(){
    $conn=connect();
    try {
        $query = $conn->prepare(
            "SELECT * FROM person WHERE name = :n and age = :a;
        ");
    }
```



```

        $query->execute(array(':n'=>$name, ':a'=>$age));
        $conn=null;
        $res=$query->fetchAll();

    }
    catch (PDOException $e) {
        echo "PDOException: ".$e->getMessage();
    }
    return $res;
}

```

From the above given code, the values for :n and :a are passed through prepared statements. This forces the developers to first define the SQL command's structure before including parameter to the query. It makes SQL aware that, whatever is passed in :n and :a, are not commands and should not be assumed as one.^[24] So, if the \$name was given an input as ' **OR 1=1;**— , the SQL would look for a name as such, instead of executing it as part of command. In this way, the SQL injection is avoided. This is the most important piece of security which cannot be overlooked while developing website with backend databases.

5.2.4 Conditions for setting Passwords

When setting passwords in Dot.Connect website, the conditions were purposely set as shown in Figure 5.7. The password must have atleast one:

The image shows a dark-themed user interface for setting a password. At the top is a light-colored rectangular input field for the password. Below it are two buttons: 'Code' and 'Mobile Number [Optional]'. To the right of the input field, a speech bubble contains the following text: 'Must contain: 1 uppercase letter, 1 lowercase letter, 1 special character, 1 numeric value and atleast 10 characters in length.' At the bottom left, there is a red asterisk followed by the text '* The first 4 credentials are must.'

FIGURE 5.7: Conditions for setting Password

- 1) uppercase letter,

- 2) lowercase letter,
- 3) special character,
- 4) number and
- 5) it had to be minimum 10 characters in length.

The reasoning behind this is, as discussed in earlier sections regarding brute force attack, character length was discussed. Suppose, the characters set was 7 word length and all in lowercase letters, the possibility of no. of words is 26 total letters in English to the power of word length, that is 26^7 in this case, which equals to 8,031,810,176 word possibility. With 4 Titan-X graphics card, which has 12 Giga onboard RAM each, hashes can be generated and compared at the rate of 40,000 words per second. But, more powerful modern crackers can crack at the rate of over 10,000,000,000 hashes per second (keeping in mind its a latest hashing algorithm). So, by having more characters sets which includes lowercase, uppercase, numbers and special characters, the character set is increased to 95. So the possibility of words to be brute forced for a 7 letter word set would be 95^7 , which totals to 69,833,729,609,375. Now, that is slightly more time consuming but is crackable in less than 2 hours.

Therefore, the minimum character length for Dot.Connect was fixed to 10. So in a best case scenario, the no. of possible words is 95^{10} , which totals to 59,873,693,923,837,890,625, making it extremely difficult to brute force as it can take almost 190 years at best (keep in mind the password length can be higher in most cases), when hashed at above mentioned rate. Consider a worst case scenario: atleast 1 letter has to be uppercase, so that is 26 letter; 1 letter has to be lowercase, so that is another 26; 1 letter has to be numerical, which is 0 to 9 totalling 10; and 1 letter is special character, which are of 33 types(including space). Assume the victim whose previous passwords were in lowercase, makes changes just enough to satisfy the above criteria. Then, the no. of possible words would be $26*26*10*33*95^6$, which totals to 163,984,298,960,625,000. Suppose a powerful system is able generate hash and compare at the rate of 10,000,000,000 words a second, it would take approximately 16,398,429 seconds or over 6 months to try on all of them, making it a humongous task.

As GPUs get faster, these barriers go down. Higher minimum character length could have been an ideal requirement, but users tend to forget passwords which are of higher

length than 10, which includes lowercase, uppercase, number and special character. There are chances when passwords are too long, users tend to store the password somewhere on the hard disk, which is also a security risk. However, with combinations of uppercase, lowercase, special character and number, the possibility of dictionary attack is decreased and also to a large extent brute force too, as explained above.

5.2.5 Minimum Password Attempt

It was significant in creating an impediment for hackers trying to brute force from the login page. So a condition was created, whereby the user who fails in attempting to login continuously for 3 times, the account gets blocked. The account has to be reactivated for the user to attempt any further logins.

Initially, there was an intention of designing the case such that the user gets blocked if he attempts to login 3 times from one particular IP address. This could create complication as the hacker can attempt to login from several systems or use VPN software that can change IP address to several fake ones with each try. Hence, the idea was dropped.

However, suppose an user attempts to login from a new IP address, he/she gets notified with a warning message to his email about the login attempt. This is the primary reason for having an extra table to store the login location details in database.

5.2.6 Hashing & Salting (cryptography)

A cryptographic hash function is a special class of hash function that has certain properties which make it suitable for use in cryptography. It is a mathematical algorithm that maps data of arbitrary size to a bit string of a fixed size (a hash function) which is designed to also be a one-way function, that is, a function which is infeasible to invert. The only way to recreate the input data from an ideal cryptographic hash function's output is to attempt a brute-force search of possible inputs to see if they produce a match, or use a rainbow table of matched hashes.^[25]

Previously there have been many hash functions that have been used and are now easily cracked, like MD5 and SHA-1. There are websites that still use MD5 algorithm

unfortunately. These functions have been around for so long, so its no longer termed safe. For Dot.Connect website, SHA-512 was used, which is one of the latest version. When brute-forcing, SHA-512 with its complicated algorithm and 512-bit length hash (which is really big), can help in lowering the speed of the brute force even more.

There is a technique which complements hashing and is called salting. Salt is a randomly generated string of characters. The primary function of salts is to defend against its hashed equivalent, or a pre-computed rainbow table attack[26]. The generated salt is then used to concatenate it with the password and then hash it. For Dot.Connect the following code was used to generate salts for passwords:

```
function salt(){
    return substr(hash('sha256', rand()), 13, 25);
}
```

The salt and password were added together and has was generated for the final combination using the following code:

```
function createhash($pass){
    return substr(hash('sha512', $pass), 20, 45);
}
```

Finally, the salt and hash was stored in the database as shown in Figure 5.8. So when confirmation was required whether the username and password are matching, the user provided password is concatenated with salt from database, the combination is further hashed and compared with hash stored in database to confirm.

regNum	name	email	hash	salt	token	active
123	Arjun KP	ak750@kent.ac.uk	ff76798fc4209d99b65aad77d0417ef00494d9f57dee	a339ab0ealfb1a3d4a217ble7	189706333674688	1
123	Arjun KP	ak75000@kent.ac.uk	721081644ef733412771d921d672c7420b932da6e828c	3800c275f83df9017f180b67c	330820009498900	1
543543435	Mithun K P	mits@driks.com	1afcdf975f6caf90b46ac8c93012467ff8271dcfdbf46	c8398d6841539b7a20ac2eaaad	808416301186446	1

FIGURE 5.8: Database depicting storage of hash and salt

5.2.7 Usage of Tokens

Since the Dot.Connect website is partly about account creation, there was need to handle account activation carefully. For account to be activated, confirmation was

required if the email is genuine. The program would send an activation mail which includes a link in the content. This link has 2 elements passed to it, namely, email and token as shown in Figure 5.9. The process of account activation is explained in previous chapter in detail

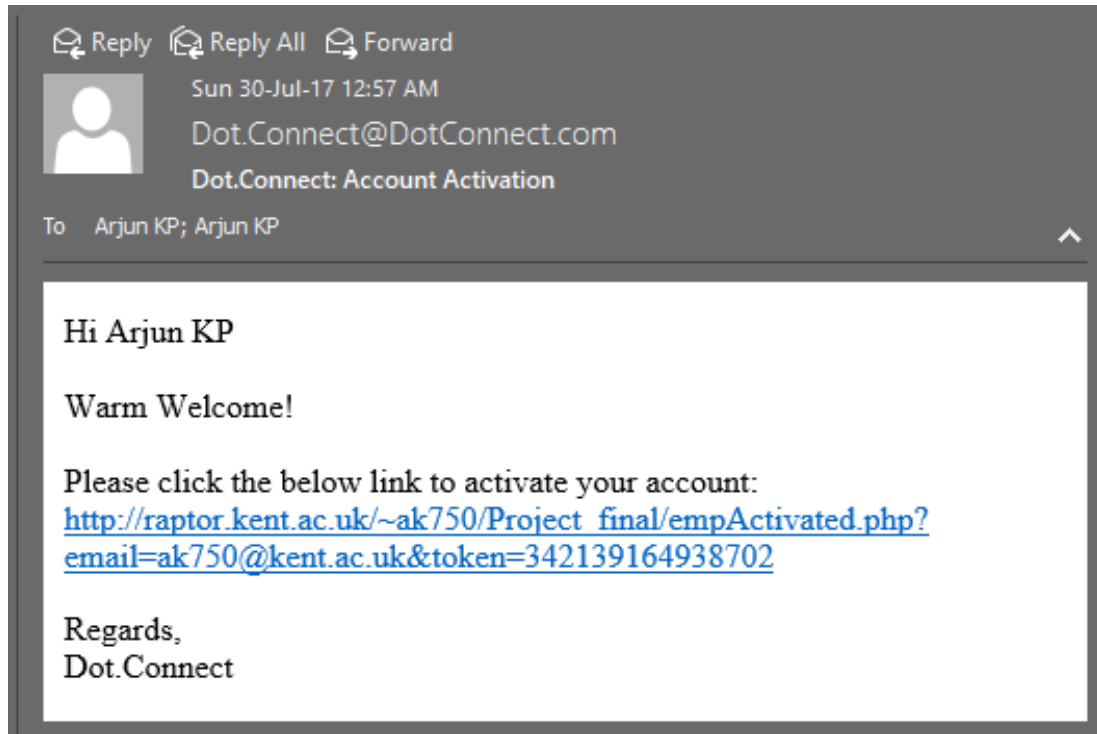


FIGURE 5.9: Email for account activation

Suppose, the account gets blocked and is tendered inactive for some reason, the user could reuse the same token to activate, which is not advisable. So, once the account is activated, a new token is generated and updated in the database. Also, each time the user requests for account activation, a new token is again generated and stored before sending the same to the mail for activation. The token updation process has been implemented on all processes, from where token is sent to mail. The same technique is followed for mobile verification as well, where the tokens are sent via SMS. For the website, cryptographically secure token generator have been used, so the user will not be able to guess the next token, at any point.

With this, the approach and measures taken to prevent hackers from transgressing into the database comes to an end. In the following chapters, explanation of the kinds of tests conducted, followed by critical analysis of the project has been illustrated.

Chapter 6

Testing & Scrutiny

This chapter gives a brief assessment of the testing method conducted on the user input, along with overall analysis of the project including the shortcomings. Discussion on whether all the objectives were met, reasons behind them are brought to light. This part of the development is an important factor as this basically sets tone for what are the changes that can be made and what are the mistakes that can be avoided in the future expansion of the project.

6.1 Black Box Testing

In order to perform black box testing, a switching from being a programmer mode to an user mode was needed, as it required an unbiased analysis. Numerous test cases were set for webpages, where the user inputs were necessary. For each tests, based on the input, the outcome was recorded, along with remarks on whether desirable output was obtained.

The Figure 6.1, 6.2 and 6.3 exhibits few of the many test cases, that was performed. They were test cases for Company Registration process. As indicated, The receiving of activation emails was also taken care, as per the result of the process, which was crucial. On the whole, there were 7 webpages on which the tests were conducted and all their respective results has been provided in detail in Appendices A, B, C, D, E, F, G. There was atleast one instance where the glitch was found in the system from the tests, which will be discussed in the next section.

	Test Case 1 - Assume Registration Number is unique					
	Input	Register Success	Register Error	Activation Email Received	Remark	Test Success
First Name	Arjun	Yes	None	Yes	Registration was successful - Obtained Desired Output	Yes
Last Name	KP					
Company Title	BRAM					
Registration Number	123					
Company Email	ak750@kent.ac.uk					

FIGURE 6.1: Company Registration Test: Assume Registration Number is unique

	Test Case 4 - Invalid Company Title					
	Input	Register Success	Register Error	Activation Email Received	Remark	Test Success
First Name	Arjun	NO	Alert Message - Company Title not Valid	NO	Registration was NOT successful - Obtained Desired Output	Yes
Last Name	KP					
Company Title	BRAM(@@					
Registration Number	123					
Company Email	ak750@kent.ac.uk					

FIGURE 6.2: Company Registration Test: Invalid Company Title

	Test Case 7 - First Name blank					
	Input	Register Success	Register Error	Activation Email Received	Remark	Test Success
First Name		NO	Alert Message - Has to fill all fields	NO	Registration was NOT successful as first name was blank- Obtained Desired Output	Yes
Last Name	KP					
Company Title	BRAM					
Registration Number	123					
Company Email	ak750@kent.ac.uk					

FIGURE 6.3: Company Registration Test: Blank First Name

6.2 Post-Mortem

The project on the whole worked perfectly. The user was able to register his company. Following that, the employees also were able to register their account with us. With activation of the account confirmed, the employees were able to log in to their respective website as well.

Several careful assessment were made on the project and tried our best to find flaws and limitations of the web service. There were several defects, some of which were corrected, while the rest of them had to be put aside mainly because of the time constraint.

6.2.1 Password Conditions

One technique hackers use to crack passwords is by trying to concatenate company name, or first name or last name of the user with certain string combinations. The hackers usually have algorithms ready which when passed with certain string values, the program generates possible string combinations with slight tweaks to them, thereby a greater possibility of guessing users password. To avoid this, conditions should have

been set that the user cannot include company name, or his name while setting password combinations.

6.2.2 Company Website Specification

Once the employee logs in successfully, he is redirected to his company website. However, this part of the project is not well defined. For this project, 'www.' was prepended to the domain of the email provided by the user as demonstrated in the following code:

```
var domain = user.val().substring(user.val().indexOf('@')+1);  
$('#form99').attr('action', 'https://www.'+domain);
```

When employee logs in, he is redirected to the website that is outcome of the concatenated string. In most cases, its NOT a webpage exclusive to the company employees. So this had to be corrected. This is no doubt, a naive approach to deal with the issue as this is an assumption of what the URL could be. There are several ways in which this could have been handled efficiently. During the company registration process, company website URL could have been extracted from the user. But the issue with using that approach is, suppose the user grants URL information which ultimately leads to a site that has nothing to do with company and may be lead to something more serious like malware attacks, then it completely defeats the purpose of this web service. So, there would be a need for a procedure that verifies the URL is genuine.

6.2.3 Data transfer

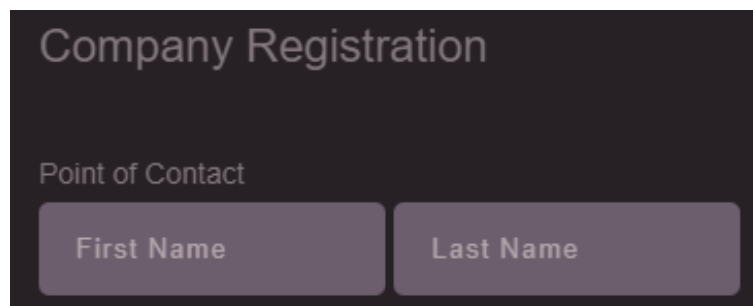
It is prerogative that the user details is passed on to the respective website with successful user login, which can include user ID and session ID. It is not always the case that the company websites may require only user ID and the session ID. There are chances, the website requires more information, and each company may have different requirements. There was a need to handle this mechanism very carefully, as these information are crucial to the respective website. Hence, this part of the project was left out. So, there wasn't anything sent in the POST method, once the user logs in, which in reality is a naive approach.

6.2.4 Company Detail Updation

Another bug which was encountered was the updation of company details part. Anyone who has the email information of the Point of Contact, can without any hurdles, use the email information to update the details such as, name of the Point of Contact person, and the Company Title. Verification could have been done by sending a token through GET method in URL form to Point of Contact email, which when clicked on, could have updated the details, instead of directly changing the details.

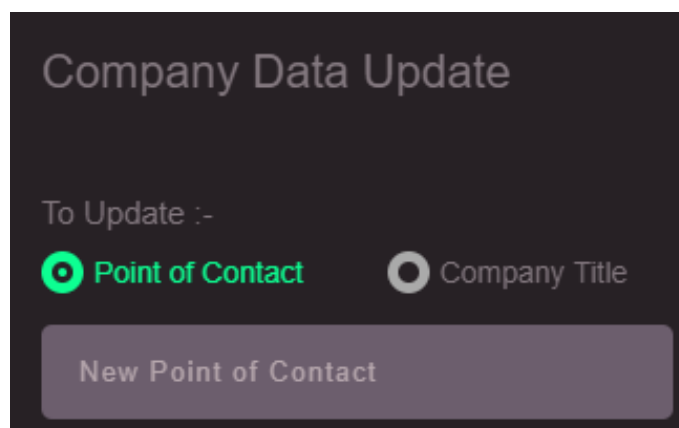
6.2.5 Point Of Contact update

One of the design problems was inconsistency in structure of data collection. For Example, during the company registration process, the name inputs of the Point of Contact was divided into First name and Last name as shown in Figure 6.4. However, when the user requests for updation of Point of Contact details, the data is collected as a single entity, not separately as shown in Figure 6.5.



The image shows a dark-themed form titled "Company Registration". Below the title, there is a label "Point of Contact". Underneath this label, there are two separate input fields: "First Name" on the left and "Last Name" on the right.

FIGURE 6.4: Company Registration: Point of Contact input in 2 separate entity



The image shows a dark-themed form titled "Company Data Update". Below the title, there is a label "To Update :-". Underneath this label, there are two radio button options: "Point of Contact" (which is selected, indicated by a green dot) and "Company Title". Below these options, there is a single input field labeled "New Point of Contact".

FIGURE 6.5: Company Detail Updation: Point of Contact input as single entity

6.2.6 No Freemails rule - not rigidly enforced

During the Company Registration process, when the user is typing his email address, which is used for company verification, there is a suggestion box that appears on the right as shown in Figure 6.6. It suggests that user should not provide any freemails, rather his/her official email ID. This part is quite crucial because, from the email, the domain is extracted and the information is used to verify whether the employees, who register for our service, are all from their respective companies they work for, and are not bogus members. Suppose, users register with freemails like gmail, they can register anyone, as almost everyone seems to have gmail or they can easily create one. Domains are kept unique to the company. Therefore, official email IDs were crucial for registration.



FIGURE 6.6: Suggestion Box for emails during company registration

Objective was to rigidly enforce the rule, which was not done. Gmail, yahoo, hotmail and outlook were added in the freemail list, that are excluded during registration. But, that was extended the list to all the freemails, which runs into hundreds of them. This was the one issue that was encountered during black box testing for company registration. The Figure 6.7, 6.8 shows different results for gmail domain and AOL domain, both of which are domains of freemail service.

Test Case 12 - Gmail as Company Email						
	Input	Register Success	Register Error	Activation Email Received	Remark	Test Success
First Name	Arjun	NO	Alert Message - Email must be company mail	NO	Registration was NOT successful as invalid company mail was blank- Obtained Desired Output	Yes
Last Name	KP					
Company Title	BRAM					
Registration Number	123					
Company Email	arjun@gmail.com					

FIGURE 6.7: Test result when using Gmail as Official mail ID

Test Case 14 - AOL mail as Company Email						
	Input	Register Success	Register Error	Activation Email Received	Remark	Test Success
First Name	Arjun	YES	None	YES	Registration was successful as AOL was not included in freemail category during coding- DID NOT Obtain Desired	No
Last Name	KP					
Company Title	BRAM					
Registration Number	123					
Company Email	arjun@aol.com					

FIGURE 6.8: Test result when using AOL freemail as Official mail ID

6.2.7 Unit Testing not done

One important mistake made was avoiding Test Driven Development. Black box testing, though significant, should not have been the only testing methods. A web service technology such as Dot.Connect, involving user authentication, required unit testing at each stage to find any issues with the code. Both PHP and JavaScript had unit testing framework namely PHPUnit and Karma respectively, which could be put to use.

6.2.8 Cohesion and Coupling issues

Though design of web page did go well, there were a lot of things that could have been better behind the scenes, i.e, on code. Whenever, there was a need to make any changes to common designs for the website in HTML, changes had to be followed with all webpages with common designs. This has no remedy in HTML by itself. `require_once()` function from PHP Could have been used by writing the common HTML codes in one PHP file and use them over and over again in every page where the design was necessary. But by the time remedy was found, 2/3rd of project was done with and with not much time left, was difficult to refactor from there.

During the design stage, the use of CSS elements appropriately was not planned. The CSS codes, though on a separate file, had elements of many web pages in a single file. Every HTML file depended on, if not all, atleast majority of the CSS files for its design elements. Ideally, the common elements could have been in one single CSS file, the rest with individual CSS files separately for individual page needs.

Apart from these, the naming of unique IDs and class names in HTML could also have been better planned and better utilized.

The JavaScript codes had similarity in few of the functions in certain files when using blur and in-focus elements of CSS, which also could have been handled as a common element and reused. The PDO used for connecting to SQL was very repetitive. Though the codes weren't all that similar, the common elements could have been separated out and made a single function that handled all kinds of SQL commands.

With this the project analysis comes to an end. Next chapter will concentrate on what can be done to improve the project in the future.

Chapter 7

Conclusion

The objectives of the project were met as expected. Whatever was envisioned in the initial stage, it was executed with satisfactory output. The usage of frameworks like jQuery and also Ajax made life easier.

On the whole, inspite of the drawbacks mentioned in previous chapter, project was satisfactory. The objectives were well defined and clear. The project implementation was executed as per the plan. However, the strategies for the implementation could have been better, but that's for another day. The final product was competent and reasonable.

7.1 Future Developments

There certainly exists huge scope for further development of the project, by extending, as per the needs of the user. Therefore, user experience and user feedback can be crucial in most future additions and changes to the software. Also, in this chapter, improvements and reforms are highlighted.

7.1.1 Structural Development

In terms of structure of the project, lot of refactoring is necessary, whether in HTML, CSS, PHP or JavaScript as explained in previous chapter. More effort is required

to improve the coupling and cohesion issues. The application of Ajax could also be extended to all 3 processes, enriching user experience even more.

7.1.2 Service on Mobile Platform

Project could be extended to mobile platform as well. Instead of having separate design for mobile, a technique called responsive web design can be implemented, so that there is no need to worry about developing design elements for mobile platform. By implementing the service in multiple platforms, especially by making it available in play store and app store, user experience can be made into a business strategy.

7.1.3 Better use of Data Collection

During the Employee Registration process, mobile information was collected from the user. Though its an optional element and there was no use of the information in any form in the project, it was still verified by sending a token to the user. Since, the effort was made, there is a need to upgrade the service. Mobile details can be utilized when user requests for password reset, which for some users could be more convenient than using email.

7.1.4 reCAPTCHA

The utilization of reCAPTCHA is an significant factor especially for login process as this can help curb hacking softwares from wielding means by which it can attack the login webpage. This feature was supposed to be added in the project but finally went ahead with decision to not implement the feature.

7.1.5 Extra Features

As per the needs, more features can certainly be added to the web service. Once the user logs in, the system could be extended by handling the domain service as well. The company information, the company web page design and themes specific to the company, can also be serviced from our side. This can further enhance the scope of the project multifold. By entering domain service market, there will be competition with

other brands by providing cheaper quality service. This way, capital can be generated for the services offered.

7.1.6 Customer Feedback

As explained earlier, customer feedback can be pivotal to improving the service and technology further. Therefore, a feature could be added where the user can post messages to the admin to share views about the software and how it can be further enhanced.

Appendix A

Black Box Test Results - Company Registration

	Test Case 1 - Assume Registration Number is unique					
	Input	Register Success	Register Error	Activation Email Received	Remark	Test Success
First Name	Arjun	Yes	None	Yes	Registration was successful - Obtained Desired Output	Yes
Last Name	KP					
Company Title	BRAM					
Registration Number	123					
Company Email	ak750@kent.ac.uk					

	Test Case 2 - Invalid First Name					
	Input	Register Success	Register Error	Activation Email Received	Remark	Test Success
First Name	Arjun""\	NO	Alert Message - Name not Valid	NO	Registration was NOT successful - Obtained Desired Output	Yes
Last Name	KP					
Company Title	BRAM					
Registration Number	123					
Company Email	ak750@kent.ac.uk					

	Test Case 3 - Invalid Last Name					
	Input	Register Success	Register Error	Activation Email Received	Remark	Test Success
First Name	Arjun	NO	Alert Message - Name not Valid	NO	Registration was NOT successful - Obtained Desired Output	Yes
Last Name	KP*&*&^^					
Company Title	BRAM					
Registration Number	123					
Company Email	ak750@kent.ac.uk					

	Test Case 4 - Invalid Company Title					
	Input	Register Success	Register Error	Activation Email Received	Remark	Test Success
First Name	Arjun	NO	Alert Message - Company Title not Valid	NO	Registration was NOT successful - Obtained Desired Output	Yes
Last Name	KP					
Company Title	BRAM(@@					
Registration Number	123					
Company Email	ak750@kent.ac.uk					

	Test Case 5 - Invalid E-mail					
	Input	Register Success	Register Error	Activation Email Received	Remark	Test Success
First Name	Arjun	NO	Alert Message - Invalid Email	NO	Registration was NOT successful - Obtained Desired Output	Yes
Last Name	KP					
Company Title	BRAM					
Registration Number	123					
Company Email	ak750@cc					

Test Case 6 - Assume Previously used Registration Number						
	Input	Register Success	Register Error	Activation Email Received	Remark	Test Success
First Name	Arjun	NO	Error Message - Company already resgitered	NO	Registration was NOT successful as the provided registration number already exists- Obtained Desired Output	Yes
Last Name	KP					
Company Title	BRAM					
Registration Number	123					
Company Email	ak750@kent.ac.uk					

Test Case 7 - First Name blank						
	Input	Register Success	Register Error	Activation Email Received	Remark	Test Success
First Name		NO	Alert Message - Has to fill all fields	NO	Registration was NOT successful as first name was blank- Obtained Desired Output	Yes
Last Name	KP					
Company Title	BRAM					
Registration Number	123					
Company Email	ak750@kent.ac.uk					

Test Case 8 - Last Name blank						
	Input	Register Success	Register Error	Activation Email Received	Remark	Test Success
First Name	Arjun	NO	Alert Message - Has to fill all fields	NO	Registration was NOT successful as last name was blank- Obtained Desired Output	Yes
Last Name						
Company Title	BRAM					
Registration Number	123					
Company Email	ak750@kent.ac.uk					

Test Case 9 - Company Title blank						
	Input	Register Success	Register Error	Activation Email Received	Remark	Test Success
First Name	Arjun	NO	Alert Message - Has to fill all fields	NO	Registration was NOT successful as company title was blank- Obtained Desired Output	Yes
Last Name	KP					
Company Title						
Registration Number	123					
Company Email	ak750@kent.ac.uk					

Test Case 10 - Registration Number blank						
	Input	Register Success	Register Error	Activation Email Received	Remark	Test Success
First Name	Arjun	NO	Alert Message - Has to fill all fields	NO	Registration was NOT successful as registration number was blank- Obtained Desired Output	Yes
Last Name	KP					
Company Title	BRAM					
Registration Number						
Company Email	ak750@kent.ac.uk					

Test Case 11 - Company Email blank						
	Input	Register Success	Register Error	Activation Email Received	Remark	Test Success
First Name	Arjun	NO	Alert Message - Has to fill all fields	NO	Registration was NOT successful as company email was blank- Obtained Desired Output	Yes
Last Name	KP					
Company Title	BRAM					
Registration Number	123					
Company Email						

Test Case 12 - Gmail as Company Email						
	Input	Register Success	Register Error	Activation Email Received	Remark	Test Success
First Name	Arjun	NO	Alert Message - Email must be company mail	NO	Registration was NOT successful as invalid company mail was blank- Obtained Desired Output	Yes
Last Name	KP					
Company Title	BRAM					
Registration Number	123					
Company Email	arjun@gmail.com					

Test Case 13 - Assume Email Domain was already registered						
	Input	Register Success	Register Error	Activation Email Received	Remark	Test Success
First Name	Arjun	NO	Error Message - Email Domain already registered	NO	Registration was NOT successful as email domain was already used by another company- Obtained Desired Output	Yes
Last Name	KP					
Company Title	BRAM					
Registration Number	123					
Company Email	arjun@kent.ac.uk					

Test Case 14 - AOL mail as Company Email						
	Input	Register Success	Register Error	Activation Email Received	Remark	Test Success
First Name	Arjun	YES	None	YES	Registration was successful as AOL was not included in freemail category during coding- DID NOT Obtain Desired	No
Last Name	KP					
Company Title	BRAM					
Registration Number	123					
Company Email	arjun@aol.com					

Appendix B

Black Box Test Results - Company Account Activation

Test Case 1 - Assume Email is registered already					
	Input	Activation Email Received	Activation Error	Remark	Test Success
Registered Email	ak750@kent.ac.uk	Yes	None	Email Sent with activation link - Obtained Desired Output	Yes

Test Case 2 - Invalid Email					
	Input	Activation Email Received	Activation Error	Remark	Test Success
Registered Email	ak750*^^	NO	Alert Message - Invalid Email provided	Email Not Sent - Obtained Desired Output	Yes

Test Case 3 - Blank Email					
	Input	Activation Email Received	Activation Error	Remark	Test Success
Registered Email		NO	Alert Message - Don't leave fields empty	Email Not Sent - Obtained Desired Output	Yes

Test Case 4 - Assume the email ID is not registered					
	Input	Activation Email Received	Activation Error	Remark	Test Success
Registered Email	silver@driks.com	NO	Error Message - Email not registered	Email Not Sent - Obtained Desired Output	Yes

Appendix C

Black Box Test Results - Company Details Updation

Test Case 1 - Assume Email is registered already					
	Input	Updation Success	Updation Error	Remark	Test Success
To Update (Click)	Point Of Contact	Yes	None	Updation was successful - Obtained Desired Output	Yes
Point Of Contact (to update)	Arjun K Prasad				
Registered Email	ak750@kent.ac.uk				

Test Case 2 - Invalid POC entry					
	Input	Updation Success	Updation Error	Remark	Test Success
To Update (Click)	Point Of Contact	NO	Alert Message - Please correct invalid entries	Updation was NOT successful - Obtained Desired Output	Yes
Point Of Contact (to update)	Arjun*&^\$@				
Registered Email	ak750@kent.ac.uk				

Test Case 3 - Blank POC entry					
	Input	Updation Success	Updation Error	Remark	Test Success
To Update (Click)	Point Of Contact	NO	Alert Message - Don't leave fields empty	Updation was NOT successful - Obtained Desired Output	Yes
Point Of Contact (to update)					
Registered Email	ak750@kent.ac.uk				

Test Case 4 - Invalid Company Title					
	Input	Updation Success	Updation Error	Remark	Test Success
To Update (Click)	Company Title	NO	Alert Message - Please correct invalid entries	Updation was NOT successful - Obtained Desired Output	Yes
Company Title (to update)	BRAM Films*&^\$@				
Registered Email	ak750@kent.ac.uk				

Test Case 5 - Blank Company Title					
	Input	Updation Success	Updation Error	Remark	Test Success
To Update (Click)	Company Title	NO	Alert Message - Don't leave fields empty	Updation was NOT successful - Obtained Desired Output	Yes
Company Title (to update)					
Registered Email	ak750@kent.ac.uk				

Test Case 6 - Not Clicked anything to Update					
	Input	Updation Success	Updation Error	Remark	Test Success
To Update (Click)		NO	Alert Message - Please select what to update	Updation was NOT successful - Obtained Desired Output	Yes
NULL (to update)	Arjun				
Registered Email	ak750@kent.ac.uk				

Test Case 7 (when clicked on POC)- Assume that the email was not registered					
	Input	Updation Success	Updation Error	Remark	Test Success
To Update (Click)	Point Of Contact	NO	Error Message - The Email provided is not registered	Updation was NOT successful - Obtained Desired Output	Yes
Point Of Contact (to update)	Mithun				
Registered Email	mithun@kent.ac.uk				

Test Case 8 (when clicked on Company Title)- Assume that the email was not registered					
	Input	Updation Success	Updation Error	Remark	Test Success
To Update (Click)	Company Title	NO	Error Message - The Email provided is not registered	Updation was NOT successful - Obtained Desired Output	Yes
Company Title (to update)	Dhritraksh				
Registered Email	mithun@kent.ac.uk				

Test Case 9 - Email left blank					
	Input	Updation Success	Updation Error	Remark	Test Success
To Update (Click)	Company Title	NO	Alert Message - Don't leave fields empty	Updation was NOT successful - Obtained Desired Output	Yes
Company Title (to update)	Dhritraksh				
Registered Email					

Test Case 10 - Email left blank					
	Input	Updation Success	Updation Error	Remark	Test Success
To Update (Click)	Point Of Contact	NO	Alert Message - Don't leave fields empty	Updation was NOT successful - Obtained Desired Output	Yes
Point Of Contact (to update)	Dhritraksh				
Registered Email					

Test Case 11 - Invalid Email					
	Input	Updation Success	Updation Error	Remark	Test Success
To Update (Click)	Point Of Contact	NO	Alert Message - Invalid Email provided	Updation was NOT successful - Obtained Desired Output	Yes
Point Of Contact (to update)	Dhritraksh				
Registered Email	sasas\$\$\$&*				

Test Case 12 - Invalid Email					
	Input	Updation Success	Updation Error	Remark	Test Success
To Update (Click)	Company Title	NO	Alert Message - Invalid Email provided	Updation was NOT successful - Obtained Desired Output	Yes
Company Title (to update)	Dhritraksh				
Registered Email	Kyyyn&%%\$#				

Appendix D

Black Box Test Results - Employee Registration

Test Case 1 - Assume Registration Number is registered, Email Domain matches the company, but Official Email ID is NOT registered							
	Input	Register Success	Register Error	Activation Email Received	SMS Received	Remark	Test Success
First Name	Arjun	Yes	None	Yes	Yes	Registration was successful - Obtained Desired Output	Yes
Last Name	KP						
Registration Number	123						
Official Email	ak750@kent.ac.uk						
Set Password	Hjnn&h6666						
Country Code	44						
Mobile Number	7752511531						

Test Case 2 - Invalid First Name							
	Input	Register Success	Register Error	Activation Email Received	SMS Received	Remark	Test Success
First Name	Arjun^^^&*&	NO	Alert Message - Invalid First Name	NO	NO	Registration was NOT successful - Obtained Desired Output	Yes
Last Name	KP						
Registration Number	123						
Official Email	ak750@kent.ac.uk						
Set Password	Hjnn&h6666						
Country Code	44						
Mobile Number	7752511531						

Test Case 3 - Blank First Name							
	Input	Register Success	Register Error	Activation Email Received	SMS Received	Remark	Test Success
First Name		NO	Alert Message - Invalid First Name	NO	NO	Registration was NOT successful - Obtained Desired Output	Yes
Last Name	KP						
Registration Number	123						
Official Email	ak750@kent.ac.uk						
Set Password	Hjnn&h6666						
Country Code	44						
Mobile Number	7752511531						

Test Case 4 - Invalid Last Name							
	Input	Register Success	Register Error	Activation Email Received	SMS Received	Remark	Test Success
First Name	Arjun	NO	Alert Message - Invalid Last Name	NO	NO	Registration was NOT successful - Obtained Desired Output	Yes
Last Name	KP"'''-"))						
Registration Number	123						
Official Email	ak750@kent.ac.uk						
Set Password	Hjnn&h6666						
Country Code	44						
Mobile Number	7752511531						

	Test Case 12 - Password length less than 10 (Assume Registration Number is registered, Email ID is unique, Email Domain matches as well)						
	Input	Register Success	Register Error	Activation Email Received	SMS Received	Remark	Test Success
First Name	Arjun	NO	Alert Message - Password must contain atleast 1 uppercase, 1 lowercase, 1 special character, 1 number and minimum 10 letters	NO	NO	Registration was NOT successful - Obtained Desired Output	Yes
Last Name	KP						
Registration Number	123						
Official Email	ak750@kent.ac.uk						
Set Password	Hij77^%						
Country Code	44						
Mobile Number	7752511521						

Test Case 20 - Lowercase missing in password (Assume Registration Number is registered, Email ID is unique, Email Domain matches as well)							
	Input	Register Success	Register Error	Activation Email Received	SMS Received	Remark	Test Success
First Name	Arjun	NO	Alert Message - Password must contain atleast 1 uppercase, 1 lowercase, 1 special character, 1 number and minimum 10 letters	NO	NO	Registration was NOT successful - Obtained Desired Output	Yes
Last Name	KP						
Registration Number	123						
Official Email	ak750@kent.ac.uk						
Set Password	hhjhjh&^5556						
Country Code	44						
Mobile Number	7752511531						

Appendix E

Black Box Test Results - Employee Account Activation

Test Case 1 - Assume Email is registered already					
	Input	Activation Email Received	Activation Error	Remark	Test Success
Registered Email	ak750@kent.ac.uk	Yes	None	Email Sent with activation link - Obtained Desired Output	Yes

Test Case 2 - Invalid Email					
	Input	Activation Email Received	Activation Error	Remark	Test Success
Registered Email	ak750*^^	NO	Alert Message - Invalid Email provided	Email Not Sent - Obtained Desired Output	Yes

Test Case 3 - Blank Email					
	Input	Activation Email Received	Activation Error	Remark	Test Success
Registered Email		NO	Alert Message - Don't leave fields empty	Email Not Sent - Obtained Desired Output	Yes

Test Case 4 - Assume the email ID is not registered					
	Input	Activation Email Received	Activation Error	Remark	Test Success
Registered Email	silver@driks.com	NO	Error Message - Email not registered	Email Not Sent - Obtained Desired Output	Yes

Appendix F

Black Box Test Results - Employee Login

Test Case 1 - Assume Company is registered & active, Email is registered and active, and credentials also match					
	Input	Login Success	Login Error	Remark	Test Success
Registered Email	ak750@kent.ac.uk	Yes	None	Login was successful - Obtained Desired Output	Yes
Password	HJHiiu%^%666				

Test Case 2 - Assume Company is registered & active, Email is registered and active, but credentials does NOT match					
	Input	Login Success	Login Error	Remark	Test Success
Registered Email	ak750@kent.ac.uk	NO	Error Message - Credentials does NOT match	Login was NOT successful - Obtained Desired Output	Yes
Password	JkkHiiu%^%6				

Test Case 3 - Assume Company is registered & active, Email is registered and active, but credentials does NOT match for 3rd time					
	Input	Login Success	Login Error	Remark	Test Success
Registered Email	ak750@kent.ac.uk	NO	Error Message - Account Blocked, Reactivate to try again	Login was NOT successful - Obtained Desired Output	Yes
Password	JkkHiiu%^%6				

Test Case 4 - Assume Company is registered & active, Email is registered but not active					
	Input	Login Success	Login Error	Remark	Test Success
Registered Email	ak750@kent.ac.uk	NO	Error Message - Email account is inactive	Login was NOT successful - Obtained Desired Output	Yes
Password	JkkHiiu%^%6				

Test Case 5 - Assume Company is registered & active, but Email ID is not registered					
	Input	Login Success	Login Error	Remark	Test Success
Registered Email	ak750@kent.ac.uk	NO	Error Message - Email not registered	Login was NOT successful - Obtained Desired Output	Yes
Password	JkkHiiu%^%6				

Test Case 6 - Assume Company is registered but inactive					
	Input	Login Success	Login Error	Remark	Test Success
Registered Email	ak750@kent.ac.uk	NO	Error Message - Company account not activated	Login was NOT successful - Obtained Desired Output	Yes
Password	JkkHiiu%^%6				

Test Case 7 - Assume Company is not registered					
	Input	Login Success	Login Error	Remark	Test Success
Registered Email	ak750@kent.ac.uk	NO	Error Message - Company not registered	Login was NOT successful - Obtained Desired Output	Yes
Password	JkkHiiu%^%6				

Test Case 8 - Blank Email					
	Input	Login Success	Login Error	Remark	Test Success
Registered Email		NO	Alert Message - Invalid Email	Login was NOT successful - Obtained Desired Output	Yes
Password	JkkHiiu%^%6				

Test Case 9 - Invalid Email					
	Input	Login Success	Login Error	Remark	Test Success
Registered Email	ak7\$\$\$@l	NO	Alert Message - Invalid Email	Login was NOT successful - Obtained Desired Output	Yes
Password	JkkHiiu%^%6				

Appendix G

Black Box Test Results - Employee Password Reset

Test Case 1 - Assume Email is registered already					
	Input	Activation Email Received	Activation Error	Remark	Test Success
Registered Email	ak750@kent.ac.uk	Yes	None	Email Sent with activation link - Obtained Desired Output	Yes

Test Case 2 - Invalid Email					
	Input	Activation Email Received	Activation Error	Remark	Test Success
Registered Email	ak750*^^	NO	Alert Message - Invalid Email provided	Email Not Sent - Obtained Desired Output	Yes

Test Case 3 - Blank Email					
	Input	Activation Email Received	Activation Error	Remark	Test Success
Registered Email		NO	Alert Message - Don't leave fields empty	Email Not Sent - Obtained Desired Output	Yes

Test Case 4 - Assume the email ID is not registered					
	Input	Activation Email Received	Activation Error	Remark	Test Success
Registered Email	silver@driks.com	NO	Error Message - Email not registered	Email Not Sent - Obtained Desired Output	Yes

Appendix H

Choice of Programming Language

There is a lot of information as to how one can create login features for websites. However, it was important to choose what best suits the purpose.

One of the first selection was regarding the platforms and the tools. It was crucial to decide on the programming languages as it can determine how easy it is to explore its libraries and find solutions. More importantly, the security of the website built depended heavily on the language.

As far as client-side programming is concerned, the HTML and the CSS is widely used on many websites and since it can be coupled with JavaScript, it was an obvious choice. So, considerable research into formation and composition of the website were made and also took care of the choice of colours and their combination to fit into the aesthetics and artistry of the website.

Since there was a need for interactive web design for the project, the choice of JavaScript was natural. By extracting the output from HTML and CSS, and manipulate as per our needs, JavaScript becomes extremely powerful client-end tool.

The reason for opting PHP over others in terms of back end language is its library, and the amount of information and the modern frameworks available about them. To top this off, it is extremely friendly language and easy to learn and master. Most crucial

point about this language is that you can just type it into a normal text file and load into a web server.

As far as database storage and manipulation is concerned, there is no alternative to SQL. University of Kent was providing free access to MySQL through its raptor for its students, was was also used.

Bibliography

- [1] Wikipedia Document. Federated identity. URL https://en.wikipedia.org/wiki/Federated_identity.
- [2] Twitter developer documentation. URL <https://dev.twitter.com/web/sign-in>.
- [3] Facebook for developers. URL <https://developers.facebook.com/docs/facebook-login/>.
- [4] Google identity platform. URL <https://developers.google.com/identity/>.
- [5] Lwin Khin Shar and Hee Beng Kuan Tan. Defeating sql injection, 2013. URL <https://pdfs.semanticscholar.org/6092/20468b4899ce7c09243ee5aad06f0530d80b.pdf>.
- [6] PepRally. The fundamental elements of design, 2011. URL <https://vimeo.com/32944253>.
- [7] Danilo Bittorf. Newsletter sign-up page, 2012. URL <https://dribbble.com/shots/479005-Newsletter-sign-up-page>.
- [8] Umar Irshad. User interface icon - take 2, 2013. URL <https://dribbble.com/shots/1023767-User-Interface-Icon-Take-2>.
- [9] Umar Irshad. Login animation - portfolio, 2012. URL <https://dribbble.com/shots/812071-Login-Animation-Portfolio>.
- [10] Arjun K Prasad. Theatre booking system, 2017. URL <https://github.com/arjun1237/TheatreBooking---PHP-JavaScript-HTML5-CSS-and-MySQL>.

-
- [11] Jeromy French jasalguero. Secure web login example/tutorial, 2011. URL <https://stackoverflow.com/questions/6136769/secure-web-login-example-tutorial>.
 - [12] Andy Prevost Brent R. Matzelle Marcus Bointon, Jim Jagielski. Phpmailer/phpmailer. URL <https://github.com/PHPMailer/PHPMailer>.
 - [13] Text Local. Send sms. URL <http://api.txtlocal.com/docs/sendsms>.
 - [14] Neill Feather President of Sitelock. Why web security is important, 2013. URL <http://ipage.com/blog/why-web-security-is-important/>.
 - [15] Wikipedia document. Cross-site scripting, 2017. URL https://en.wikipedia.org/wiki/Cross-site_scripting.
 - [16] Wikipedia document. Sql injection. URL https://en.wikipedia.org/wiki/SQL_injection#cite_note-2.
 - [17] Dr Mike Pound. Running an sql injection attack - computerphile, 2016. URL <https://www.youtube.com/watch?v=ciNHn38EyRc>.
 - [18] Dr Mike Pound. Password cracking - computerphile, 2016. URL <https://www.youtube.com/watch?v=7U-RbOKanYs&t=20s>.
 - [19] Wikipedia document. Dictionary attack, 2017. URL https://en.wikipedia.org/wiki/Dictionary_attack.
 - [20] Openwall Document. John the ripper password cracker. URL <http://www.openwall.com/john/>.
 - [21] darknet.org. The mole automatic sql injection sqli exploitation tool, 2011. URL <https://www.darknet.org.uk/2011/12/the-mole-automatic-sql-injection-sqli-exploitation-tool/>.
 - [22] PHP documentation. stripslashes, . URL <http://php.net/manual/en/function.stripslashes.php>.
 - [23] PHP documentation. htmlspecialchars, . URL <http://php.net/manual/en/function htmlspecialchars.php>.
 - [24] Tom Scott. Hacking websites with sql injection - computerphile, 2013. URL https://www.youtube.com/watch?v=_jKylhJtPmI.

-
- [25] Wikipedia document. Cryptographic hash function, 2017. URL https://en.wikipedia.org/wiki/Cryptographic_hash_function.
- [26] Wikipedia document. Salt (cryptography). URL [https://en.wikipedia.org/wiki/Salt_\(cryptography\)](https://en.wikipedia.org/wiki/Salt_(cryptography)).