**CHAPTER 2**

**DESIGN**

**2.0 Introduction**

Systems design is the process of defining the [architecture,](https://en.wikipedia.org/wiki/Systems_architecture) modules, interfaces, and [data f](https://en.wikipedia.org/wiki/Data)or a [system t](https://en.wikipedia.org/wiki/System)o satisfy specified [requirements.](https://en.wikipedia.org/wiki/Requirement) Systems design could be seen as the application of [systems theory t](https://en.wikipedia.org/wiki/Systems_theory)[o product development.](https://en.wikipedia.org/wiki/Product_development) System design gives the entire picture of the application and interactions of its components.

**2.1 Project Plan**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PHASE** | **NAME** | **DESCRIPTION** | **START DATE** | **END DATE** |
| Phase I | Analysis | Understanding the problem, existing solution and defining the proposed Solution | 19.01.2021 | 23-02-2021 |
| Phase II | Design | Designing the project | 24-02-2021 | 02-03-2021 |
| Phase III | Implementation | Coding and  Implementing | 03-03-2021 | 19-03-2021 |
| Phase IV | Testing | Testing the system | 21-03-2021 | 30-03-2021 |
| Phase V | Documentation | Preparing the document | 01-04-2021 | 05-04-2021 |

***Table 2.1 Overall plan***

**2.2 ER diagram**

Entity Relationship model is a data model for describing the data or information aspects of a domain or its process requirements, in an abstract way that lends itself to ultimately being implemented in a database such as a relational database.

Architectural design is the high level structure. It is an early stage of the system design process. It represents the link between specification and design processes and carried out in parallel with some specification activities.

**2.2.1 Features of the ER Diagram**

* The ER model is a high-level data model diagram to represent the ER model.
* ER model is based on three basic concepts: Entities, Attributes & Relationships

**ENTITY**

**ATTRIBUTE**

**RELATIONSHIP**

***Figure 2.1 ER Diagram***

User

ADMIN

HAS

Bills

Accounts

HAS

Quick Pay

***Figure 2.2 Menu Page***

**2.3 Description of the method of solution**

The existing system does not contain the Block Chain Technology for the application with High Security and Safe Processing of the users account transaction. The previous Application and other banking application are not having the technology of the block chain and believing in the third party application for the data security. The proposed solution contains the Secure and Safe transaction for the users and Processing the application in 24\*7 at any location for safe and secure process. It also includes the database as the permeant storage in the block chain.

**2.3.1 Data flow diagram**

A Data Flow Diagram(DFD) is a graphical tool used to describe and analyse movement of data through system. It is a graphical representation of the “flow” of data through a computer system or a data or it looks at how data flows through a system. These are central tool and basic form which the other components are developed. The transformation of data from input to output, through processed may be described logically and independently of physical components associated with the system. The development of DFD is done at several levels. The flow diagram describes the boxes that describe computations, decisions, interactions & loops. It is important to keep its mind that the flow diagram are not flowchart and should not include control elements.

A Data flow diagram (DFD) is a graphical representation of the “flow” of data thought an information system, modelling its process aspects. A DFD is often is often used as a preliminary step to create an overview of the system, which can later be elaborated. DFDs can also be used for the visualization of data processing.

PROCESS

DATABASE

ATTRIBUTE

FLOW

A DFD shows what kinds of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

**DFD Level 0**

Database

User

**DFD Level 1**

Registration

Block chain

Database

Maintenance

Schedule

Previous record

Help desk

User

**DFD Level 2**

Database

Login

**DFD Level 3**

Login

Database

**2.3.2 Pseudo form**

GUI design is a type of interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation as opposed to text based interface, typed command labels or text navigation.

Home Page

Announcement

USERNAME

SERVICES

Password

Login

***Figure2.3 Homepage***

Transaction Page

Transaction Details

Account

Summary

***Figure2.4 Transaction Page***

Accounts Page

Slider 1

Account Type

Footer

***Figure2.5 Accounts Page***

Menu Page

Slider 1

User Details

Icon

Icon

Icon

Icon

Icon

Icon

Footer

***Figure2.6 Menu Page***

Admin Page

Admin

Data Record

Queries

Footer

***Figure2.7 Admin Page***

**2.3.3 Table Design**

**Table: Login\_registration**

|  |  |  |
| --- | --- | --- |
| COLUMN NAME | DATA TYPE | CONSTRAINTS |
| Name | Varchar(100) | Not Null |
| Email | Varchar(50) | Not Null |
| Account Number | Varchar(50) | Not Null |
| Branch location | Varchar(15) | Not Null |
| Username | Varchar(15) | Not Null |
| Password | Varchar(15) | Not Null |
| Confirm Password | Varchar(15) | Not Null |

***Table2.2 Login\_ registration***

**Table: Login**

|  |  |  |
| --- | --- | --- |
| COLUMN NAME | DATATYPE | DESCRIPTION |
| Username | Varchar(50) | Not Null |
| Password | Varchar(50) | Not null |

***Figure 2.3 login page***

**Table: Help\_desk**

|  |  |  |
| --- | --- | --- |
| COLUMN NAME | DATA TYPE | CONSTRAINTS |
| Name | Varchar(100) | Not Null |
| Contact | Number | Not Null |
| Email | Varchar(50) | Not Null |
| date | Date | Not Null |
| query | Varchar(50) | Not Null |

***Table 2.4 help\_desk***

**2.4 Pseudo code:**

**2.4.1 Home Page**

If the user is new user

Redirect the user to registration page

Else

Redirect to login page

**2.4.2 Registration page**

If the user doesn’t have login id

He/she has to fill up the details and register their details

Else

Redirect to the login page

**2.4.3 Login Page**

If user is approved by the database and enter details are correct

It will go to the home page

Else

Error message will be shown and farmer will reenter the login details .

**2.4.4 Admin login**

If admin entered the correct details

It will go to the main page

Else

Error message will be shown and admin will reenter the details

**2.4.5 Announcement page**

Admin will give the announcement

The details will be stored in the database and display the message in the front end

Else

If the entered details in incorrect, error message will be displayed.

**2.4.6 Request page**

User will give request for the particular things.

The request will be stored in the database and display the message to the admin

Else

If the entered details in incorrect, error message will be shown.

**2.4.7 Help page**

User will enter the details for contact

The details will be stored in the database and display the message to the viewers

Else

Error message will be shown.

**2.5 System description**

The system after careful analysis has been identified that there are four main modules and each modules comprises of several sub modules. The main modules are as follows,

* **Admin**
* **User**

**Admin**

The Admin Module is the module designed for the system administrator. The Admin is the one who controls the entire activities of the application. Admin can modify the users and grant access rights to them. The interface of the admin is totally different of that of the other user. The Admin can view all the details and activities about the labs. And also can generate report of the details.. The following are the functional activities of Admin Module:

1. **Approve/Disapprove:** Admin can approve/disapprove the registration of users.
2. **Modify, Add and Remove Users:** Admin can modify existing users, add new users and remove a user if the lab is no longer existing
3. **View Details:** Admin can view farmer details & query Details.
4. **View / Update / Delete Items:** Admin can view inventory items from all farmers and also manage the items easily.
5. **View / Update Requirements:** Admin can view and update inventory items from all farmers.
6. **Notify ERP:** Admin notify requirements in query from all farmers.
7. **Feedback:** Admin can also see the feedback on farmer’s requirements.

**User**

The menu module is the user of the system. The user logs into the system and performs various tasks. From marking queries to updating programs, requirements the new technologies various operations that reflects in other modules. The following are the functional activities of Transaction Module:

1. **Login:** The Farmers need to login using Farmer login credentials in order to access the application.
2. **Update Profile:** The Farmer can update their profiles, change passwords etc.
3. **Edit Items:** Farmers can edit apparatus requirements.
4. **View / Update Items:** Farmerscan view, update apparatus requirements

**2.6 Hardware Requirements**

The Hardware requirements many serve as the basis for a contract for the implementation of the system and should, therefore, be a complete and consistent specification of the whole system.

**Component Configuration**

|  |  |  |
| --- | --- | --- |
|  | **Component** | **Configuration** |
| Monitor |  | 15’’ LED |
| System |  | Pentium Dual Core |
| Processor |  | I5, Intel Core 8th Gen |
| RAM |  | 4 GB |
| Hard Disk |  | 120 GB and Above |
| System Type |  | 64-bit operating system |
| Input Devices |  | Keyboard , Mouse |

***Table 2.5 Hardware Requirements***

**2.7 Software Requirements**

Software requirement includes both definition and specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost,planning team activities, performing tasks throughout the development activity.

|  |  |
| --- | --- |
| Specifications | Functional Requirements |
| Database | MYSQL |
| Programming Language | PYTHON |
| Operating System | Windows 7, 8 , and new Version |
| Tool | Pycharm , Anaconda |

***Table2.6 Software Requirements***

**PyCharm :**

PyCharm is a hybrid-platform developed by JetBrains as an IDE for Python. It is commonly used for Python application development. Some of the unicorn organizations such as Twitter, Facebook, Amazon, and Pinterest use PyCharm as their Python IDE! It supports two versions: v2.x and v3.x.

PyCharm is [cross-platform](https://en.wikipedia.org/wiki/Cross-platform), with [Windows](https://en.wikipedia.org/wiki/Windows), [macOS](https://en.wikipedia.org/wiki/MacOS" \o "MacOS) and [Linux](https://en.wikipedia.org/wiki/Linux) versions. The Community Edition is released under the [Apache License](https://en.wikipedia.org/wiki/Apache_License), and there is also Professional Edition with extra features – released under a [proprietary license](https://en.wikipedia.org/wiki/Proprietary_software).

Coding assistance and [analysis](https://en.wikipedia.org/wiki/Code_analysis), with [code completion](https://en.wikipedia.org/wiki/Autocomplete), syntax and error highlighting, [linter integration](https://en.wikipedia.org/wiki/Lint_(software)), and quick fixes

Project and code navigation: specialized project views, file structure views and quick jumping between files, classes, methods and usages

Python [refactoring](https://en.wikipedia.org/wiki/Refactoring): includes rename, extract method, introduce variable, introduce constant, pull up, push down and others

Version control integration: unified user interface for [Mercurial](https://en.wikipedia.org/wiki/Mercurial), [Git](https://en.wikipedia.org/wiki/Git_(software)), [Subversion](https://en.wikipedia.org/wiki/Apache_Subversion), [Perforce](https://en.wikipedia.org/wiki/Perforce) and [CVS](https://en.wikipedia.org/wiki/Concurrent_Versions_System) with change lists and merge

Support for scientific tools like matplotlib, numpy and scipy [professional edition only.

It competes mainly with a number of other Python-oriented IDEs, including [Eclipse](https://en.wikipedia.org/wiki/Eclipse_(software))'s [PyDev](https://en.wikipedia.org/wiki/PyDev" \o "PyDev), and the more broadly focused [Komodo IDE](https://en.wikipedia.org/wiki/Komodo_IDE).

**ANACONDA**

Anaconda is a distribution of the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) and [R](https://en.wikipedia.org/wiki/R_(programming_language)) programming languages for [scientific computing](https://en.wikipedia.org/wiki/Scientific_computing) ([data science](https://en.wikipedia.org/wiki/Data_science), [machine learning](https://en.wikipedia.org/wiki/Machine_learning) applications, large-scale data processing, [predictive analytics](https://en.wikipedia.org/wiki/Predictive_analytics), etc.), that aims to simplify [package management](https://en.wikipedia.org/wiki/Package_management) and deployment.

The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc., which was founded by Peter Wang and [Travis Oliphant](https://en.wikipedia.org/wiki/Travis_Oliphant) in 2012. As an Anaconda, Inc. product, it is also known as Anaconda Distribution or Anaconda Individual Edition, while other products from the company are Anaconda Team Edition and Anaconda Enterprise Edition, both of which are not free.

Package versions in Anaconda are managed by the [package management system](https://en.wikipedia.org/wiki/Package_manager) *[conda](https://en.wikipedia.org/wiki/Conda_(package_manager)" \o "Conda (package manager))*.This package manager was spun out as a separate open-source package as it ended up being useful on its own and for other things than Python.

There is also a small, bootstrap version of Anaconda called Miniconda, which includes only conda, Python, the packages they depend on, and a small number of other packages.

Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from [PyPI](https://en.wikipedia.org/wiki/Python_Package_Index" \o "Python Package Index) as well as the [conda](https://en.wikipedia.org/wiki/Conda_(package_manager)" \o "Conda (package manager)) package and virtual environment manager. It also includes a GUI, Anaconda Navigator,as a graphical alternative to the command line interface (CLI).

The big difference between conda and the [pip package manager](https://en.wikipedia.org/wiki/Pip_(package_manager)) is in how package dependencies are managed, which is a significant challenge for Python data science and the reason conda exists.

In contrast, conda analyses the current environment including everything currently installed, and, together with any version limitations specified (e.g. the user may wish to have Tensorflow version 2,0 or higher), works out how to install a compatible set of dependencies, and shows a warning if this cannot be done.

**Blockchain**

A blockchain, originally block chain is a growing list of [records](https://en.wikipedia.org/wiki/Record_(computer_science)), called blocks, that are linked using [cryptography](https://en.wikipedia.org/wiki/Cryptography). Each block contains a [cryptographic hash](https://en.wikipedia.org/wiki/Cryptographic_hash_function) of the previous block,[[6]](https://en.wikipedia.org/wiki/Blockchain#cite_note-cryptocurrencytech-6) a [timestamp](https://en.wikipedia.org/wiki/Trusted_timestamping), and transaction data (generally represented as a [Merkle tree](https://en.wikipedia.org/wiki/Merkle_tree)). By design, a blockchain is resistant to modification of its data. This is because once recorded, the data in any given block cannot be altered retroactively without alteration of all subsequent blocks.

For use as a distributed ledger, a blockchain is typically managed by a [peer-to-peer](https://en.wikipedia.org/wiki/Peer-to-peer) network collectively adhering to a [protocol](https://en.wikipedia.org/wiki/Protocol_(communication)) for inter-node communication and validating new blocks. Although blockchain records are not unalterable, blockchains may be considered [secure by design](https://en.wikipedia.org/wiki/Secure_by_design) and exemplify a distributed computing system with high [Byzantine fault tolerance](https://en.wikipedia.org/wiki/Byzantine_fault_tolerance). The blockchain has been described as "an open, [distributed ledger](https://en.wikipedia.org/wiki/Distributed_ledger) that can record [transactions](https://en.wikipedia.org/wiki/Financial_transaction) between two parties efficiently and in a verifiable and permanent way".

 The identity of Satoshi Nakamoto remains unknown to date. The invention of the blockchain for bitcoin made it the first digital currency to solve the [double-spending](https://en.wikipedia.org/wiki/Double-spending) problem without the need of a trusted authority or central [server](https://en.wikipedia.org/wiki/Server_(computing)).

Logically, a blockchain can be seen as consisting of several layers.

* infrastructure (hardware)
* [networking](https://en.wikipedia.org/wiki/Network_layer) (node discovery, information propagation and verification)
* [consensus](https://en.wikipedia.org/wiki/Consensus_(computer_science)) ([proof of work](https://en.wikipedia.org/wiki/Proof_of_work), [proof of stake](https://en.wikipedia.org/wiki/Proof_of_stake))
* data (blocks, transactions)
* [application](https://en.wikipedia.org/wiki/Application_layer) ([smart contracts](https://en.wikipedia.org/wiki/Smart_contract)/[dApps](https://en.wikipedia.org/wiki/Decentralized_application" \o "Decentralized application), if applicable).

**2.5 Conclusion**

This chapter started with project plan, followed by database schema representation, respected diagrams and software & hardware requirements along with the descriptions. The overall design structure of the project is also portrayed. The following chapter focuses on the implementation phase.

**CHAPTER 3**

**IMPLEMENTATION**

**3.0 Introduction**

Implementation is the stage of the project when the theoretical design turned into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve the changeover and evaluation of changeover methods.

**3.1 Method of solution related to problem**

A web app is an application that the user does not download and instead accesses via a web browser over a network. Example web browsers include Google Chrome, Safari and Mozilla Firefox. While there are many banking applications without data security setting for the user. It does not contain any previous records for the better understanding of the user. The best solution is to fix the data lose problem by data security to the registered user’s mobile phone and to store the data records in the application. It also updated whenever needed. The queries raised by the user is also cleared by help desk simultaneously.

**3.2 Accurate method of solution**

The accurate method of solution links the objective with the implementation process. It includes how each module works its functionality. The modules are integrated properly to make communication of data to be done properly. The application works in a dynamical and also it fetch the result for the user in a fast manner. It is the solution for the manual system that is being computerized. There are some other methods were used to solve this solution or problem. The objectives of the project have been analysed and accomplished. Accurate method of solution meets the requirements and has done with it. The proposed system has the following modules Storing the data records of all the data permantely, can be updated and solves the complaints or queries which was raised by the user.

Fixing the reminder to the registered user’s mobile phone for updates and news.

**3.2.1 XRP server Connection**

The XRP server is a digital asset built for payments. It is the native digital asset on the XRP Ledger—an open-source, permissionless and decentralized blockchain technology that can settle transactions in 3-5 seconds. XRP can be sent directly without needing a central intermediary, making it a convenient instrument in bridging two different currencies quickly and efficiently. Faster, less costly and more scalable than any other digital asset, XRP and the XRP Ledger are used to power innovative technology across the payments space.

**3.2.2 Message Verification**

The Interactive messaging is fixed for reminding the users about the transaction done by the user. Users register their information upon desired application. The Interactive messaging works like the notification. It notifies a time before the transaction gets completed with verification. While registering, the pop-up message will pop in the client’s phone. Automatically the message will be saved in the phone’s storage.

**3.2.3 Data Storage**

The year’s data records were stored in the application for the clear understanding of the user. It can be updated whenever needed. The help desk process for the user, it includes the user’s complaint and FAQS about the Transaction. The storage will save all the transaction record in the block chain permantely and separating the session as the time and date with the transaction record , the separation of the block is done in the transaction done in the particular time and saves once’s all the process have been completed .

**3.2.4 Help Desk**

The Helpdesk will store all the queries and problems of the user in a record were stored in the application for the clear understanding of the Banking . It can be updated the query details onces it is viewed whenever needed. The help desk process for the user, it includes the user’s complaint and FAQS about the Transaction. Admin will rectify the user’s complaints and FAQS for the users. The complaint is delivered to the admin in the form of pop-up message and official email. The response from the admin should be delivered within twenty-four hours of the complaint. The complaint and FAQS is fixed in separate roots in the admin’s application.

**3.3 Conclusion**

Once the implementation phase is done, it is left to the developer or the testing team to check if each functionality of the application for which the code was developed and its works fine.

The various text are carried out are discussed in the next chapter.

**CHAPTER 4**

**TESTING AND DOCUMENTATION**

**4.0 Introduction**

Testing is one of the process by which one detects the defects in the software. It is considered as the final opportunity for developing team to defect and convert or rectify any defecting that may appear during the software development stage. Software testing is a process of testing a program with the explicit intention of review in software products and related documentation for completion, correction, reliability and maintainability.

Testing Objectives are,

* All field entries must work properly
* Ensure database is maintained with integrity and confidential
* The entry screen, messages and responses must be validated
* Validation of input has been done
* No duplicate entries should be allowed
* All links should take the user to the correct page .

**4.1 Testing Strategies**

The purpose of testing is to discover uncovered errors. Testing is the process of trying to discover every faults or weakness in the products. It provides a way to check the functionality of components, sub-assemblies. It is the process of exercising software with intent of ensuring that the software system meets its requirements and user exceptions and does not fail in an unacceptable manner. There are various types of test. Each type of test addresses specific testing requirements. Testing is the process in which the software tester has knowledge of the inner working structure and language of the software. The following are the types of testing that have been done to check and overcome the errors in the web application.

* Unit Testing
* Integration Testing
* Database Testing

**4.1.1 Test Cases**

A test case is an item which contains the following information

A set of test inputs. These are data items received from external source by the code under test. The external source can be hardware, software or human.

Execution conditions. These are conditions required for running the test, for examples, a certain state of database, or configuration of hardware device.

Expected output. These are the specified results to be produced by the code under test.

**4.1.2 Test Plan**

Test plan describes an overall strategy and a procedure that defines specific steps and the tests that will be conducted. An effective test plan and procedure will lead to the orderly construction of the software and the discovery of errors at each stage in the construction process.

**4.2 Test Results**

Testing strategy leads to test results. Actual testing will take place in this section. Test results describes an input, action or an Expected output, to determine if a feature of an application is working correctly and come up with an output testing phase. As mentioned in the previous section the test deliverables such as test cases, Project/Module name, test case description. Test input, expected output, actual output are contained in the test case. Every module was tested under certain conditions and errors were rectified. The following modules are tested and the actual results are furnished below.

* Registration
* Data Storage

**4.2.1 Unit Testing**

This type of testing is performed by developers before the setup is handed over to the testing team to formally execute the test cases. Unit testing is performed by the respective developers on the individual units of source code assigned areas. The developers use test data that is different from the test data of the quality assurance team. The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

**Test outline**

**Case 1:** Inter-Field dependencies

**Case 1.1:** Testing the login form, user email id and password input field, phone number and Account details.

*Figure No 4.1 Outline Iteration 1*

**Table Name: Test results for unit testing**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S. No** | **Module Name** | **Test Condition** | **Test Data/Input** | **Expected**  **Result** | **Actual**  **Result** |
| 1 | Login Module | Click on the login button after entering email id and password | Enter invalid email id & password | Message prompts that  “check username & password” | Message prompts that  “check username & password” |
| 2 | Login Module | Click on the login button after entering email-id and password | Enter valid email id & password | Message prompts that  “Successfully logged in” | Message prompts that  “Successfully logged in” |
| 3 | Registration  Module | Click on submit button after entering the details of user | Enter invalid mobile number | Message prompts that  “Mobile number should be of 10 digits” | Message prompts that  “Mobile number should be of 10 digits” |
| 4 | Registration  Module | Click on submit button after entering the  Mail id | Enter Invalid  Email id | Message prompts that  “Please check  Mail id” | Message prompts that  “Please check  Mail id ” |
| 5 | Registration  Module | Click on submit button after entering the details of user | Enter valid  Details | Message prompts that “Successfully Registered” | Message prompts that “Successfully Registered” |

*Table No 4.1 Test results for unit testing*

**Test Results**

All the test cases mentioned above passed successfully. No defects are encountered.

**4.2.2 Integration Testing**

Integration testing is defined as the testing of combined parts of an application to determine if they function correctly. Integration testing can be done in two ways they are

Bottom-up integration: This testing begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds.

Top-down integration: In this testing, the highest-level modules are tested first and progressively, lower-level modules are tested thereafter.

**Test outline**

**Case 2:** Interaction between modules of the application

**Case 2.1**: Check if login button is redirect to registration form

**Case 2.2:** Check if pressing the register button, the notification is giving the pop-up message with sound or not.

**Case 2.3:** check if all the data records are stored correctly without any duplicated data.

**Case 2.4**: Check if pressing previous records is redirects to corresponding record details page and displays the data records.

**Case 2.5:** Check if the help page form is redirecting to the particular page and  rectifying the problem or not.

*Figure No 4.2 Outline Iteration*

**Table Name: Test results for Integration testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No** | **Module**  **Name** | **Test Condition** | **Expected Result** | **Actual Result** |
| 1 | Login page | Click on login button after entering valid input data | Navigate to the  Alternate Page | Navigate to the  Alternate Page |
| 2 | Account page | Account details must be displayed with all the updates . | Navigate to the Account task  activity | Navigate to the Account task activity |
| 3 | Data  Records | Click on display button for the details of previous Transaction | Previous details should be displayed. | Previous details should be displayed in the page. |
| 4 | Help Desk | Email is received from the user and should clear the solution. | The FAQS must be  clarified by the admin and should be updated soon. | The FAQS must be clarified by the admin and should be updated soon. |

*Table No 4.2 Test results for Integration testing*

**Test Results**

All the test cases mentioned above passed successfully. No defects are encountered.

**4.2.3 Database Testing**

Database testing involves the retrieved values from the database by the web or desktop application. Data in the User Interface should be matched as per the records are stored in the database.

**Test outline**

**Case 3:** Check whether the user is allowed to perform only those specific operations which are specified by the event requirements.

**Case3.1**: Check whether the registration and entered data are correct. Also, the data are

fetched correctly.

**Case 3.2:** Check whether the details are retrieved correctly from the database.

*Figure No 4.3 Outline Iteration 3*

**Table Name: Test results for Database testing**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.No** | **Module**  **Name** | **Test Condition** | **Test Data/Input** | **Expected**  **Result** | **Actual**  **Result** |
| 1 | Registration | Click the register button to save the details in the database | Personal and details are entered | The personal and details has been successfully registered | The personal and details has been successfully registered |
| 2 | Data Storage | Data records are stored and the information should be retrieved | All data records are entered | details has been successfully fetched and displayed | details has been successfully fetched and displayed |

*Table No 4.2 Test results for Integration testing*