PROJECT REPORT

ON

Bank Management System

Submitted towards Partial Fulfilment for the Requirements of Fourth Semester of

MASTER OF COMPUTER APPLICATION

Batch : 2020-22

Submitted By

Dinanath Yadav : 2000360140040

Abhishek Thakur : 2000360140005

Gautam Dev : 2000360140046

Vikash Mishra : 200036014115

Supervised by

Mr. Aravendra Kumar Sharma

INSTITUTE OF MANAGEMENT STUDIES

GHAZIABAD UTTAR PRADESH- 201009

AFFILIATED TO DR. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW

SESSION: 2021-22

INSTITUTE OF MANAGEMENT STUDIES
GHAZIABAD, INDIA

IMS GHAZIABAD

PROJECT ACCEPTANCE CERTIFICATE

This is to certify that project work entitled **"Bank Management System"** by **Dinanath Yadav, Abhishek Thakur, Gautam Dev, Vikash Mishra** Student of **Fourth Semester in MASTER OF COMPUTER APPLICATION** of **INSTITUTE OF MANAGEMENT STUDEIS, GHAZIABAD affiliated** to **Dr. A.P.J ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW,** hereby accepted for the partial fulfilment for the requirement of the **Fourth** semester of **MASTER OF COMPUTER APPLICATION.**

Prof. (Dr.) Nripendra Dwivedi
Dean- MCA

Mr. Aravendra Kumar Sharma
Project Supervisor

DECLARATION

I, **Dinanath Yadav,** do hereby declare that the project work entitled **Bank Management System** is an authenticated work carried out by me under the guidance of **Mr. Aravendra Kumar Sharma** for the partial fulfilment for the requirement of the **Fourth** semester of **MASTER OF COMPUTER APPLICATION** and this work has not been submitted for similar purpose anywhere else except to **INSTITUTE OF MANAGEMENT STUDIES, GHAZIABAD**, affiliated to **Dr. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW.**

Dinanath Yadav

Roll No. 2000360140040

DECLARATION

I, **Abhishek Thakur** do hereby declare that the project work entitled **Bank Management System** is an authenticated work carried out by me under the guidance of **Mr. Aravendra Kumar Sharma** for the partial fulfilment for the requirement of the **Fourth** semester of **MASTER OF COMPUTER APPLICATION** and this work has not been submitted for similar purpose anywhere else except to **INSTITUTE OF MANAGEMENT STUDIES, GHAZIABAD**, affiliated to **Dr. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW.**

Abhishek Thakur

Roll No. 2000360140005

DECLARATION

I **Gautam Dev** do hereby declare that the project work entitled **Bank Management System** is an authenticated work carried out by me under the guidance of **Mr. Aravendra Kumar Sharma** for the partial fulfilment for the requirement of the **Fourth** semester of **MASTER OF COMPUTER APPLICATION** and this work has not been submitted for similar purpose anywhere else except to **INSTITUTE OF MANAGEMENT STUDIES, GHAZIABAD**, affiliated to **Dr. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW.**

Gautam Dev

Roll No. 2000360140046

DECLARATION

I **Vikash Mishra** do hereby declare that the project work entitled **Bank Management System** is an authenticated work carried out by me under the guidance of **Mr. Aravendra Kumar Sharma** for the partial fulfilment for the requirement of the **Fourth** semester of **MASTER OF COMPUTER APPLICATION** and this work has not been submitted for similar purpose anywhere else except to **INSTITUTE OF MANAGEMENT STUDIES, GHAZIABAD**, affiliated to **Dr. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW.**

Vikash Mishra

Roll No. 2000360140115

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my supervisor**, Mr. Aravendra Kumar Sharma,** for proper supervision and timely support. His / Her wide knowledge, guidance and continuous encouragements have been a great help throughout our project work.

I would like to thank **Prof. (Dr.) Nripendra Dwivedi, Dean-MCA** and all faculty members of ***Department of Computer Applications, Institute of Management Studies, Ghaziabad*** who kindly have given valuable guidelines and suggestions for completion of my project, and also motivated me throughout entire project.

Dinanath Yadav
Abhishek Thakur
Gautam Dev
Vikash Mishra

TABLE OF CONTENTS

Page no.

Introduction

1.1 Introduction:

This project intends to introduce more user friendliness in the various activities such as customer/admin login, record updation, maintenance, and searching. The searching of record has been made quite simple as all the details of the customer can be obtained by simply keying in the identification or account number of that customer. Similarly, record maintenance and updation can also be accomplished by using the account number with all the details being automatically generated. These details are also being promptly automatically updated in the master file thus keeping the record absolutely up-to-date.

Bank Management System is based on PHP and is a major project. It is used to keep the records of clients, employee etc in bank. The system provides the access to the customer to create an account, deposit/withdraw the amount from his account, also to view transactions history of all accounts present. The following presentation provides the specification for the system.

This project is provide various activities such as account management, open new account, manage account, transaction details, deposit amount, credit, withdraw and transfer amount into one account to other account.

The entire information has maintained in the database or Files and whoever wants to retrieve can't retrieve, only authorization user can retrieve the necessary information which can be easily be accessible from the file.

1.2 Objective:

A computer based management system is designed to handle all the primary information required to maintain the customer account which include the statement of any customer transaction and account details. Separate database is maintained to handle all the details required for the correct statement calculation and generation.

- To keep record updation, maintenance, and customer details.
- All information /knowledge sharing to Customer.
- Using the account number with all the details being automatically generated.
- Automatically updated in the master file thus keeping the record absolutely up-to-date.
- This system is very easy to use, so that any user can use without getting pre-knowledge about this.
- Its very much user friendly and easy to access the bank management system.
- Quick and Fast access to the database.

1.3 Need Of Project:

The bank management system is an application for maintain a person account in a bank. The system provides the access to the customer to create an account, deposit/withdraw the cash from his account, also to view reports of all accounts present. These are the needs of projects.

- Security of customer and bank information.
- Require more physical work and manpower.
- All the manual entry and editing will take more time.
- No level of clearance for the different levels of employees.
- Safety of paper documents from the disaster.
- Backup of the information.

The information will be secure from the different types of disasters as there will be an automatic backup system for the customer and bank information.

- Maintain data integrity Validate the manual calculations avoid calculation error.
- Safeguard the data accuracy.
- More reliable and efficient.
- More user-friendly interface.

Feasibility Study

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.
Three key considerations involved in the feasibility analysis are:

Economic Feasibility: This study is carried out to check the economic impact will have on the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customised products have to be purchased.

Technical Feasibility: This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes for the implementing this system.

Operational Feasibility: The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

Software Requirement Specifications

3.1 Software Functional Requirements:

There are a lot of software requirements specifications included in the functional requirements of the Bank Management System, which contains various process, namely Login, Account Management and Database.

Admin Side Functionality:

- Dashboard Page
- List of All Transactions History
- Deposit for Client
- Withdraw for Client
- Fund Transfer for Client
- Manage System Credentials
- Manage System Settings/Info

Client Side Functionality:

- Dashboard Page (display the account number and current balance)
- List of Transactions History
- Deposit
- Withdraw
- Fund Transfer
- Manage System Credentials

Public Side Functionality:

- Login Page
- Announcement Page
- About us Page

Login Process of SRS (Software Requirements Specification)

● **Adding Customer:** The Bank Management System enables the staffs in the front desk to include new Customer to the system.
● **Assigning an ID to the Customer**: The OBMS enables the staff in the front desk to provide a unique ID for each Customer and then add them to the record sheet of the Customer. The Customer can utilize the ID throughout their bank stay.

Check Out of SRS:

● **Deleting Customer ID:** The staff in the administration section of the ward can delete the Customer ID from the system when the Customer's checkout.

Report Generation of SRS:

● **Information of the Customer:** The Bank Management System generates a report on every Customer regarding various information like Customer name, Account number, Email Id and Password whom its assigns, pin, and more.

Database of SRS:

● **Mandatory Customer Information:** Every Customer has some necessary data like account number, their first, middle name and last name, email id, password and pin etc.

● **Updating information of the Customer:** The Bank Management System enables users to update the information of the Customer as described in the mandatory information included.

3.2 Specific Requirements:

3.2.1 Hardware Requirements:

Processor- Intel(R) Core(TM) i3/i5 CPU
128MB RAM or More
32/64-bit operating system
Hard disk- 160GB

3.2.2 Software Interfaces:

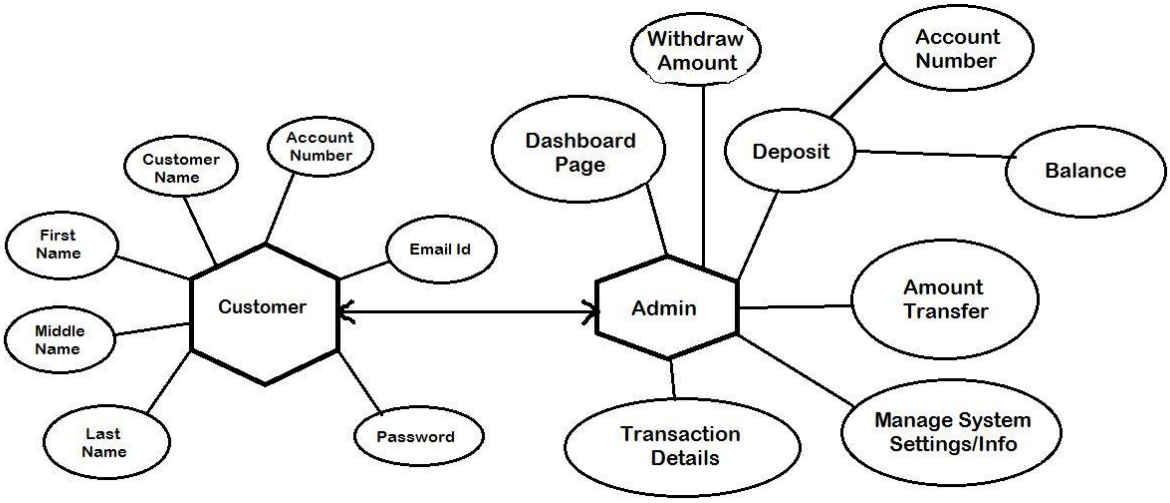
Platform used : Wampserver- 2.0
Server : Apache- 2.2.11
Back end : MySQL- 5.1.36
Front end : PHP- 5.3.0, HTML , JavaScript and CSS, Bootstrap(3.4.1)
Operating system : Windows 8 or higher.

3.3 Software Limitations:

- It is not that secure due to its open-source, because the ASCII text file are often easily available.
- It is not suitable for giant content-based web applications.
- It has a weak type, which can cause incorrect data and knowledge to user.
- The PHP frameworks aren't equivalent in behavior so does their performance and features.

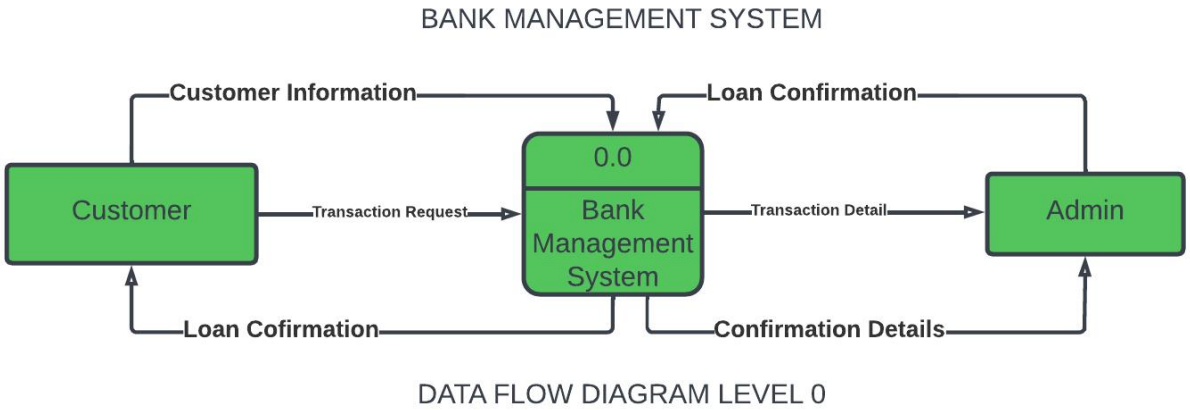
Design

4.1 ER Diagram-

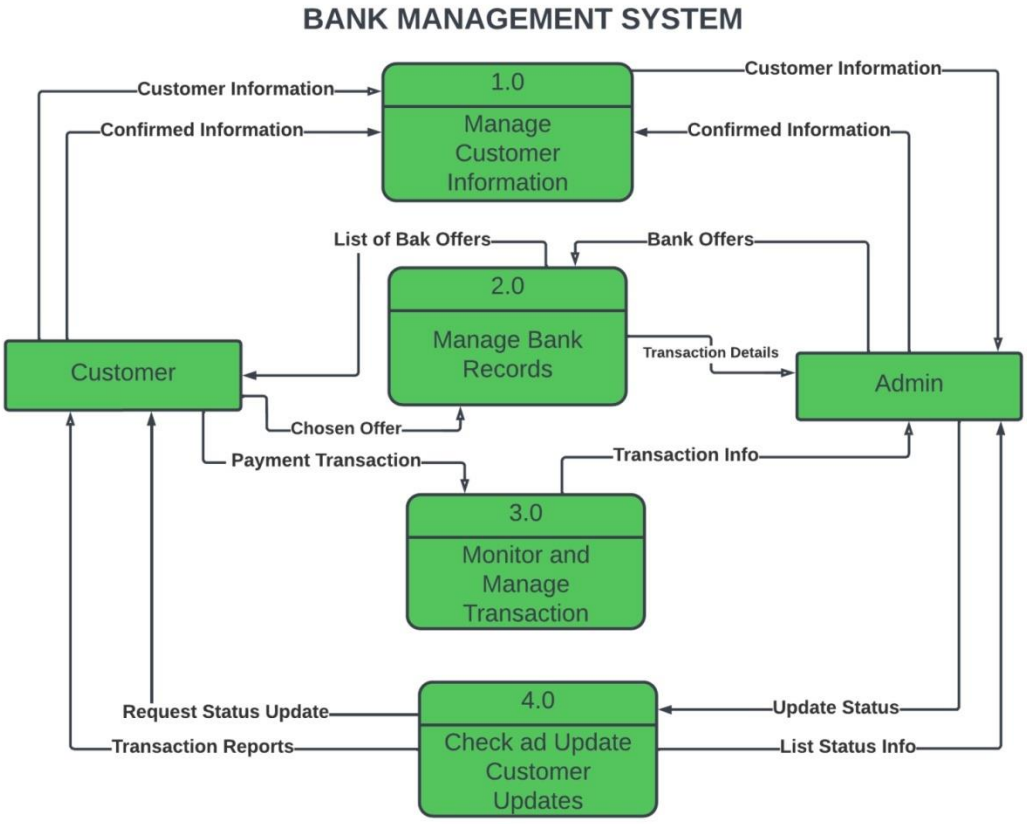


4.2 Data Flow Diagram:

Zero Level DFD:



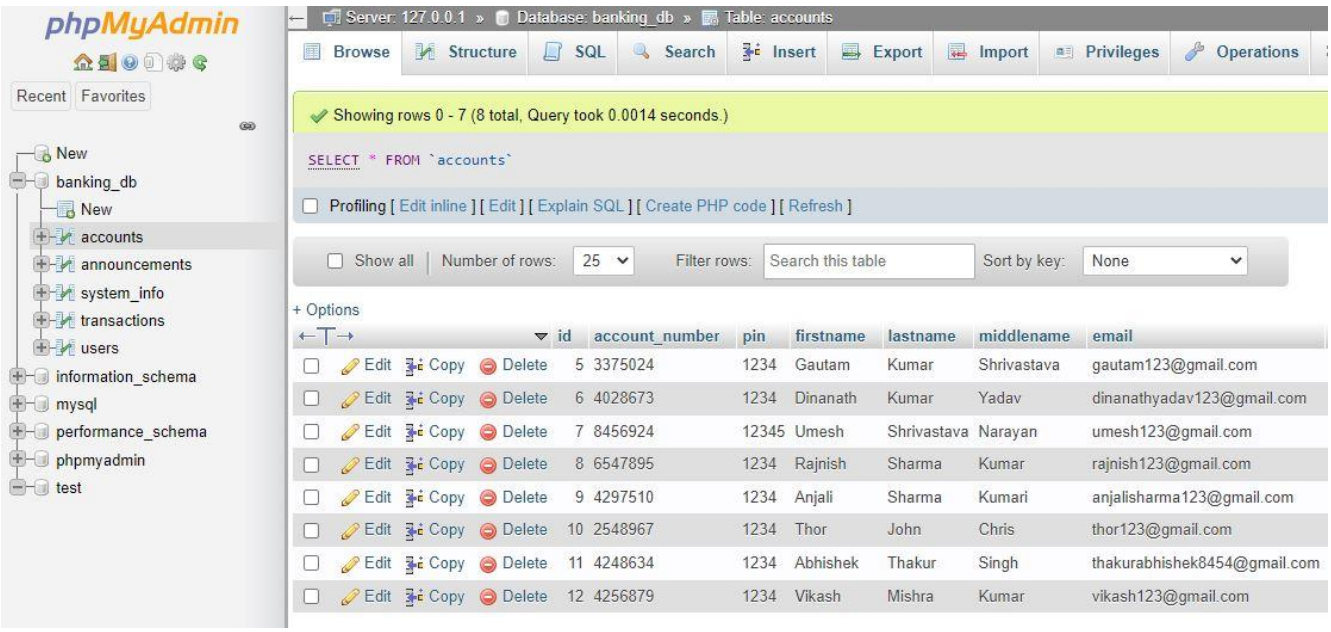
First Level DFD:



DATA FLOW DIAGRAM LEVEL 1

4.3 Database Design:

	Table	Action						Rows	Type	Collation	Size	Overhead
☐	accounts		Browse	Structure	Search	Insert	Empty	Drop	8	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	announcements		Browse	Structure	Search	Insert	Empty	Drop	3	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	system_info		Browse	Structure	Search	Insert	Empty	Drop	5	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	transactions		Browse	Structure	Search	Insert	Empty	Drop	26	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	users		Browse	Structure	Search	Insert	Empty	Drop	1	InnoDB	utf8mb4_general_ci	16.0 KiB	-
	5 tables	Sum							43	InnoDB	utf8mb4_general_ci	80.0 KiB	0 B



Testing (Test Cases & Results)

Introduction Of Testing-

Software testing is the process of evaluation a software item to detect differences between given input and expected output. Also to assess the feature of A software item. Testing assesses the quality of the product. Software testing is a process that should be done during the development process. In other words software testing is a verification and validation process.

Verification

Verification is the process to make sure the product satisfies the conditions imposed at the start of the development phase. In other words, to make sure the product behaves the way we want it to.

Validation

Validation is the process to make sure the product satisfies the specified requirements at the end of the development phase. In other words, to make sure the product is built as per customer requirements.

Basics of software testing:

There are two basics of software testing: blackbox testing and whitebox testing.

Blackbox Testing

Black box testing is a testing technique that ignores the internal mechanism of the system and focuses on the output generated against any input and execution of the system. It is also called functional testing.

Whitebox Testing

White box testing is a testing technique that takes into account the internal mechanism of a system. It is also called structural testing and glass box testing.
Black box testing is often used for validation and white box testing is often used for verification.

UNIT TESTING

Introduction:

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process. It forms the basis for component testing. Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended.

Benefits :

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. A unit test provides a strict, written contract that the piece of code must satisfy. As a result, it affords several benefits.

1) Find problems early : Unit testing finds problems early in the development cycle. In test-driven development (TDD), which is frequently used in both extreme programming and scrum, unit tests are created before the code itself is written. When the tests pass, that code is considered complete. The same unit tests are run against that function frequently as the larger code base is developed either as the code is changed or via an automated process with the build. If the unit tests fail, it is considered to be a bug either in the changed code or the tests themselves. The unit tests then allow the location of the fault or failure to be easily traced. Since the unit tests alert the development team of the problem before handing the code off to testers or clients, it is still early in the development process.

2) Facilitates Change : Unit testing allows the programmer to refactor code or upgrade system libraries at a later date, and make sure the module still works correctly (e.g., in regression testing). The procedure is to write test cases for all functions and methods so that whenever a change causes a fault, it can be quickly identified. Unit tests detect changes which may break a design contract.

3) Simplifies Integration : Unit testing may reduce uncertainty in the units themselves and can be used in a bottom-up testing style approach. By testing the parts of a program first and then testing the sum of its parts, integration testing becomes much easier.

4) Documentation : Unit testing provides a sort of living documentation of the system. Developers looking to learn what functionality is provided by a unit, and how to use it, can look at the unit tests to gain a basic understanding of the unit's interface (API). Unit test cases embody characteristics that are critical to the success of the unit. These characteristics can indicate appropriate/inappropriate use of a unit as well as negative behaviours that are to be trapped by the unit. A unit test case, in and of itself, documents these critical characteristics, although many software development environments do not rely solely upon code to document the product in development.

INTEGRATION TESTING:

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing.

Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

Purpose

The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items. These "design items", i.e., assemblages (or groups of units), are exercised through their interfaces using black-box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test whether all the components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e., unit testing. The overall idea is a "building block" approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages. Software integration testing is performed according to the software development life cycle (SDLC) after module and functional tests. The cross-dependencies for software integration testing are: schedule for integration testing, strategy and selection of the tools used for integration, define the cyclomatical complexity of the software and software architecture, reusability of modules and life-cycle and versioning management. Some different types of integration testing are big-bang, top-down, and bottom-up, mixed (sandwich) and risky-hardest. Other Integration Patterns are: collaboration integration, backbone integration, layer integration, client-server integration, distributed services integration and high-frequency integration.

Big Bang

In the big-bang approach, most of the developed modules are coupled together to form a complete software system or major part of the system and then used for integration testing. This method is very effective for saving time in the integration testing process. However, if the test cases and their results are not recorded properly, the entire integration process will be more complicated and may prevent the testing team from achieving the goal of integration testing.A type of big-bang integration testing is called "usage model testing" which can be used in both software and hardware integration testing. The basis behind this type of integration

testing is to run user-like workloads in integrated user-like environments. In doing the testing in this manner, the environment is proofed, while the individual components are proofed indirectly through their use. Usage Model testing takes an optimistic approach to testing, because it expects to have few problems with the individual components. The strategy relies heavily on the component developers to do the isolated unit testing for their product. The goal of the strategy is to avoid redoing the testing done by the developers, and instead flesh-out problems caused by the interaction of the components in the environment. For integration testing, Usage Model testing can be more efficient and provides better test coverage than traditional focused functional integration testing. To be more efficient and accurate, care must be used in defining the user-like workloads for creating realistic scenarios in exercising the environment. This gives confidence that the integrated environment will work as expected for the target customers.

Top-down And Bottom-up

Bottom-up testing is an approach to integrated testing where the lowest level components are tested first, then used to facilitate the testing of higher level components. The process is repeated until the component at the top of the hierarchy is tested. All the bottom or low-level modules, procedures or functions are integrated and then tested. After the integration testing of lower level integrated modules, the next level of modules will be formed and can be used for integration testing. This approach is helpful only when all or most of the modules of the same development level are ready. This method also helps to determine the levels of software developed and makes it easier to report testing progress in the form of a percentage. Top-down testing is an approach to integrated testing where the top integrated modules are tested and the branch of the module is tested step by step until the end of the related module. Sandwich testing is an approach to combine top down testing with bottom up testing.

Testing Result:

The reports panel has a summary showing how many tests failed, how many had errors and how many were fixed. If no comparison can be done because data for the base branch is not available, the panel will just show the list of failed tests for head.
There are four types of results:

Newly failed tests: Test cases which passed on base branch and failed on head branch

Newly encountered errors: Test cases which passed on base branch and failed due to a test error on head branch

Existing failures: Test cases which failed on base branch and failed on head branch

Resolved failures: Test cases which failed on base branch and passed on head branch.
Each entry in the panel will show the test name and its type from the list above. Clicking on the test name will open a modal window with details of its execution time and the error output.

Implementation/Technical Environment

User requirements:

The initial activity of the process model is to analyze the user requirements. It is obvious that in any software project, the first and foremost activity is to have an understanding of what the users require.
There are various tools and techniques to analyze user requirements. However, it is important to note that, while it is important to understand the user requirement in full, the idea here is not to write loads of documents, especially when it comes to PHP projects. What is more important is to make sure that we model what the users require and, based on the models, get the users to provide feedback, and then improve the model based on the user feedback. Some simple techniques that can be used to capture requirements include:

- Studying the existing system
- Observation of the users in action in a real environment
- Interviews with potential users and stakeholders of the system
- Some tools that can be used to capture requirements include:
- Data flow diagrams showing how data flows and processing happens in the system

Use case diagrams that portray how potential users of the system would be using the system. When using use case diagrams, it is critical to capture all of the major

use cases of the system at a high level. It does not need to be too detailed, but we need to ensure that we have not missed any key use case scenario in the implementation. We can always seek the help of the users to help us validate the use cases and fill in the gaps, in case we have missed any key user requirements.
When documenting user requirements, we need to be comprehensive, concise, and clear. This is because the team members need to be able to refer to those, whenever they want. If the requirement specifications are too bulky and take time to read, it might not help the team. Therefore, it is always a good idea to use diagrams and tables to summarize information, whenever possible, to make sure that the captured information can be grasped by the team members at a glance.
We need to make sure that the team members really refer to the requirements to during the project life cycle. Therefore, anything that encourages them to use those requirement specifications is welcome and would make the project's success more probable.

User activity analysis:

While executing business functions with the software application, users engage in various activities using the system. Analyzing those activities becomes the next important activity in the process, and it also opens up the doors to understanding and designing the user interface.
Based on the business model and the data model, we can analyze the various activities that the users would want to carry out with the system. We can make use of the data processing functions of data flow diagrams, or use case diagrams, and come up with a list of user activities for each business use case of the system.
The list of user activities becomes the input for the storyboard design activity.

The next activity is the implementation, keeping open the option of switching the database management systems or supporting multiple database management systems open. In this activity, we might either use the database management systems directly or use SQL and try to be database agnostic. In addition to creating the database, the team members involved with the data layer implementation also devise plans for testing and also for carrying out unit testing.
In addition to SQL based unit testing, they need to implement the database access logic, which is also a part of the database layer, using PHP. The team members can then implement unit tests with PHP to verify the implementation. Executing the unit tests can be automated using PHP frameworks, upon each source code change.

Once the implementation is complete, it can be tested and verified completely using system testing by a quality assurance team.
Database implementation is a prerequisite for business logic implementation. However, business logic design need not wait until the database implementation is complete. As soon as data model is available, business modeling can be done and followed by business logic design.

Conclusion and Future Scope

Future Scope:

Bank Management have undergone a change for its betterment. The administrations of customer sector are opting IT solutions for the better management and Customer care in their bank campus. Have a look at some salient features of bank management software.

Daily functions like Customer login, updates, managing account balance and overall management of various departments can be easily performed with higher accuracy after the installation of bank management.

The modules of online bank management software are user-friendly and easy to access. It has a common user friendly interface having several modules. The officials can utilize these modules in their processes without any hassle and make the best possible use of Bank Services.

This project can be handled in future by doing various modifications like: -

- We can go further for Online Banking.
- We can establish and start various Branches and available help centers for Account Holder's Queries.

- We can also deal through internet by creating web pages and a banking website for internet dealing.

- To attract Account Holder's we can offer various offers during festivals months.

- We can also deal in various types of Banking Transactions.

- To have more and more customer satisfaction we will emphasize more and more on our dealings.

Conclusion

A bank account is not only about saving money, it's also about managing money. Opening an account is a smart move - it means that you can access a service that helps you control your money, and which may help you borrow at some time in the future, if you need to do so. But do remember that you are the customer - that means you have rights and if you're not happy, you can complain, and you can move your account somewhere else.

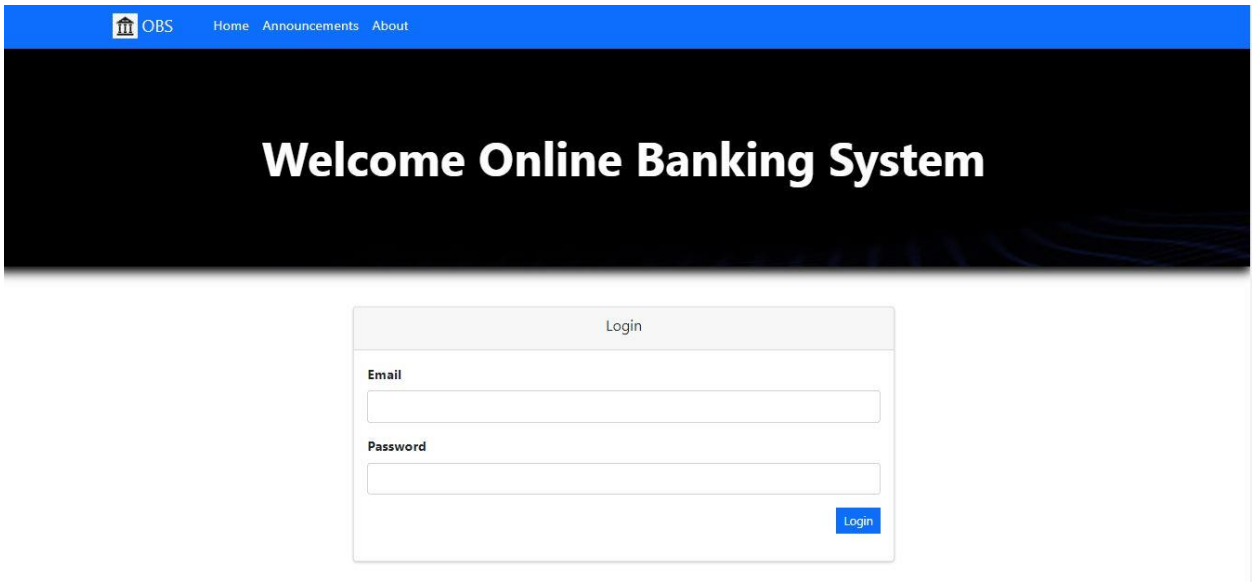
Bank management system is a virtualization of transactions in banking system. The banking system are used manual working but when we used online banking system it is totally virtualization process which avoid manual process and converts it in automatic process.

The bank management system is an application for maintain a person and customer account in a bank. The system provides the access to the customer to create an account, deposit/withdraw the cash from his account, also to view reports of all accounts present.

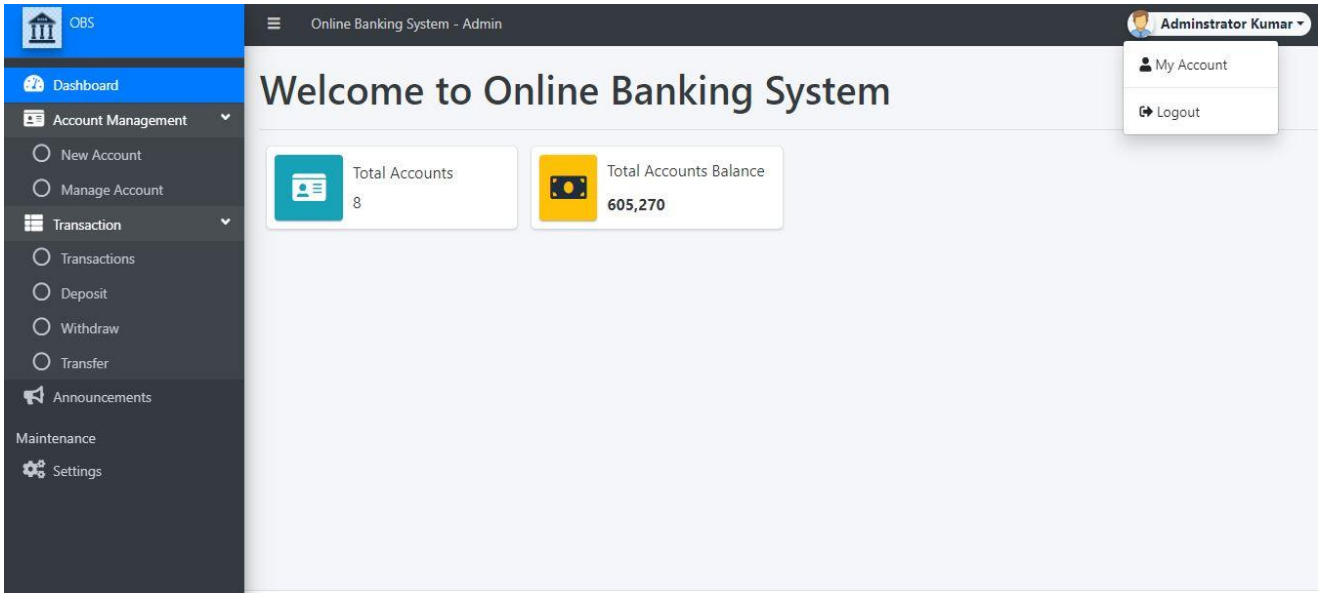
The system automates basic banking activities to aid a bank clerk's day-to-day operations. Additionally, the system's purpose is to allow for the storage of information for a large number of client accounts, as well as the ability to add, retrieve, and edit information for any account.

Input/Output Forms

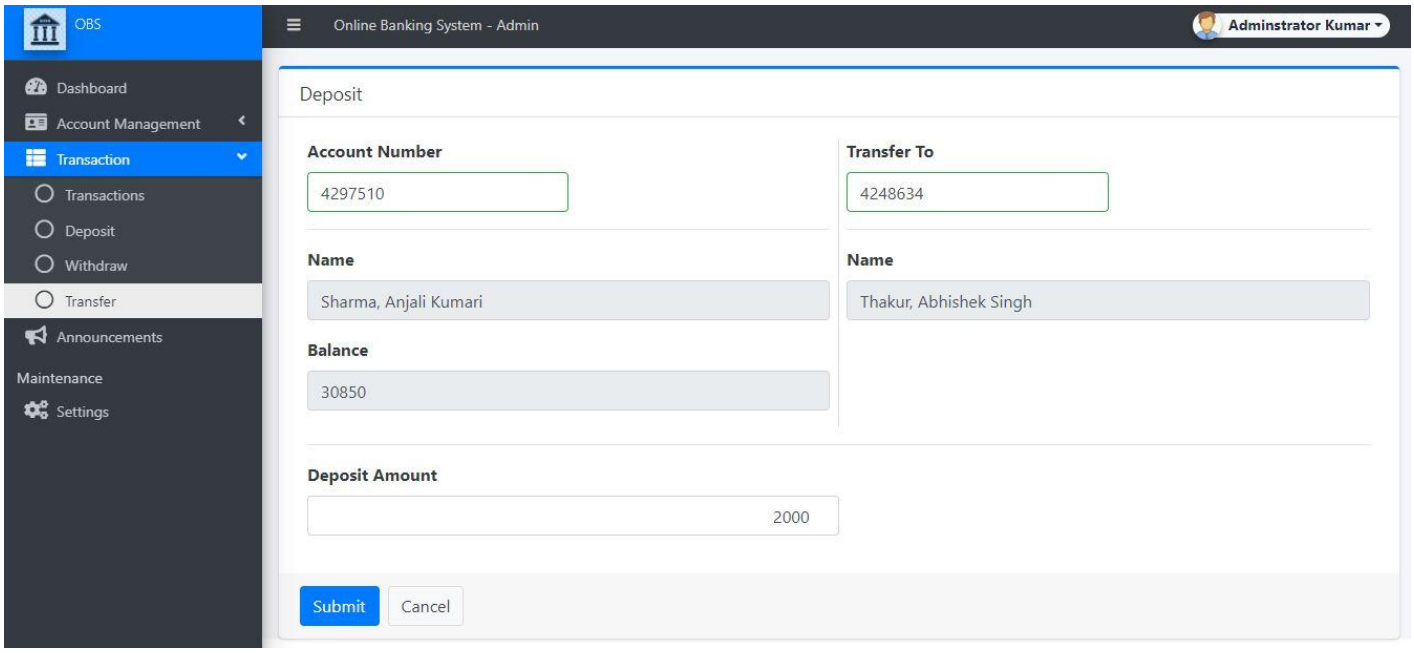
Login Page:



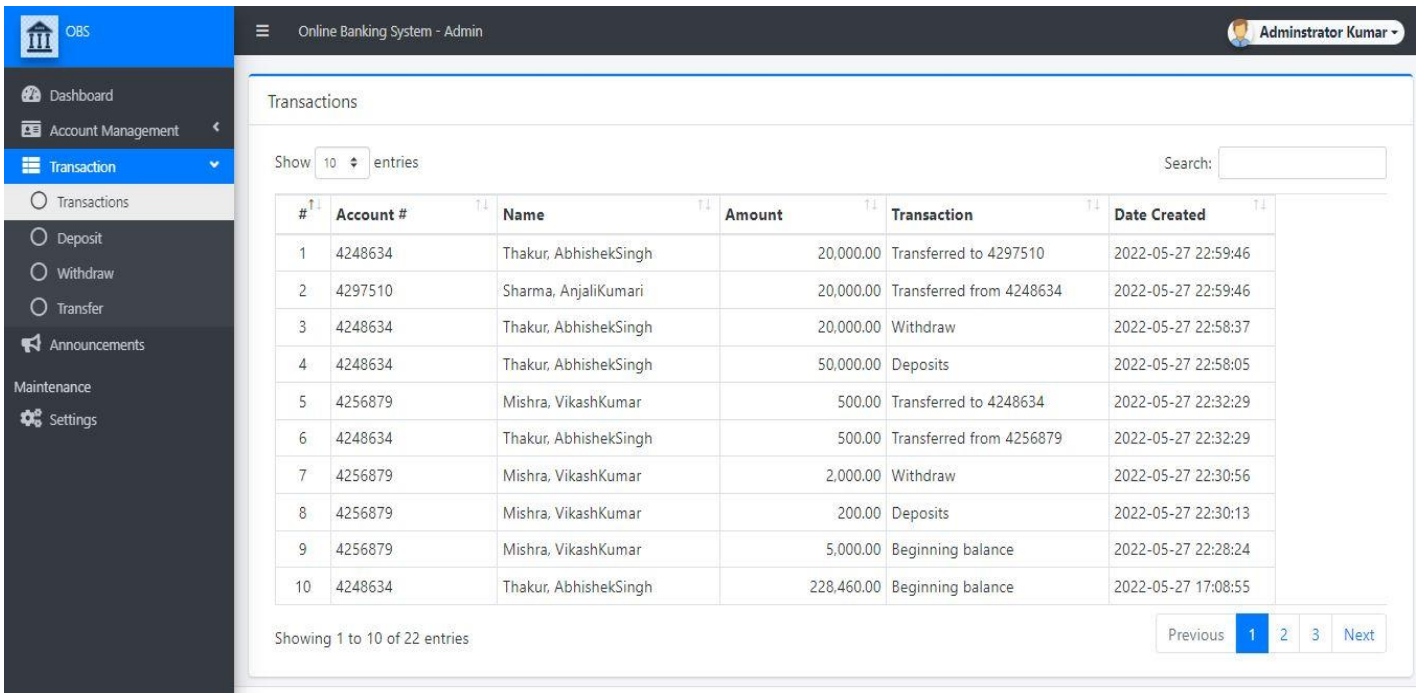
Admin Home Page:



Amount Transfer Page:

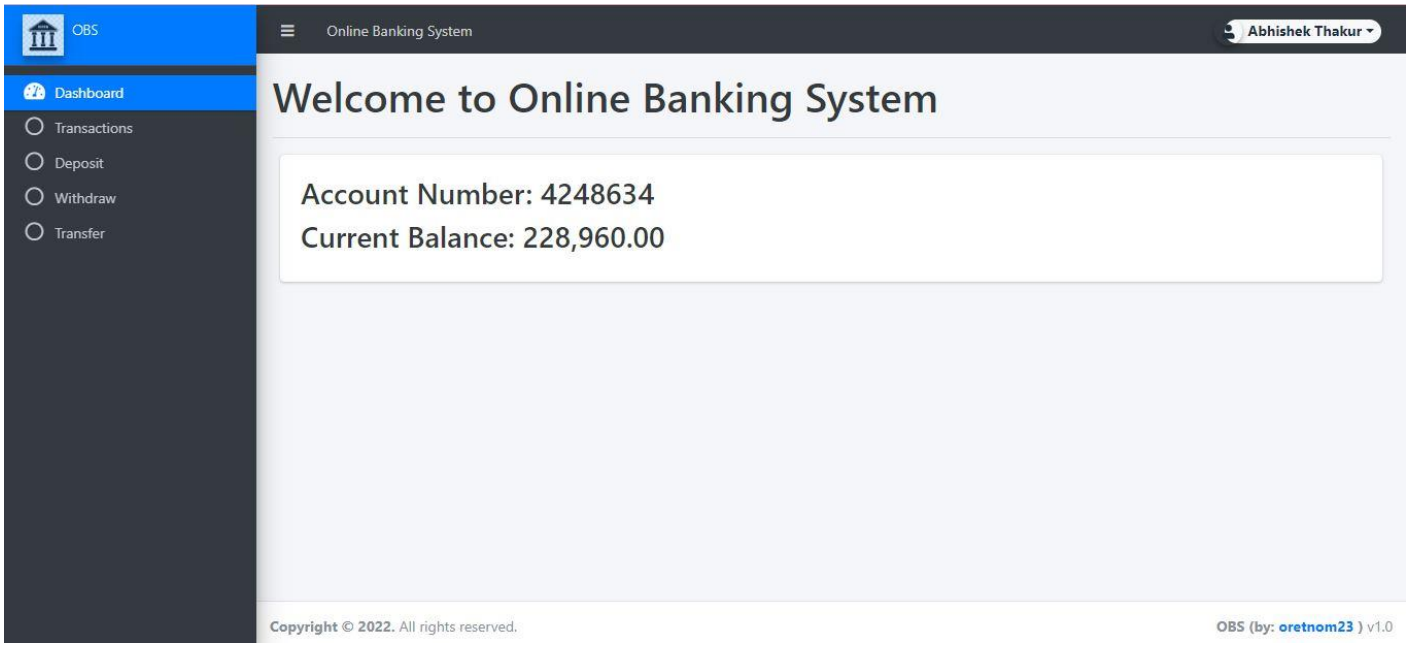


Admin Transaction List Page:

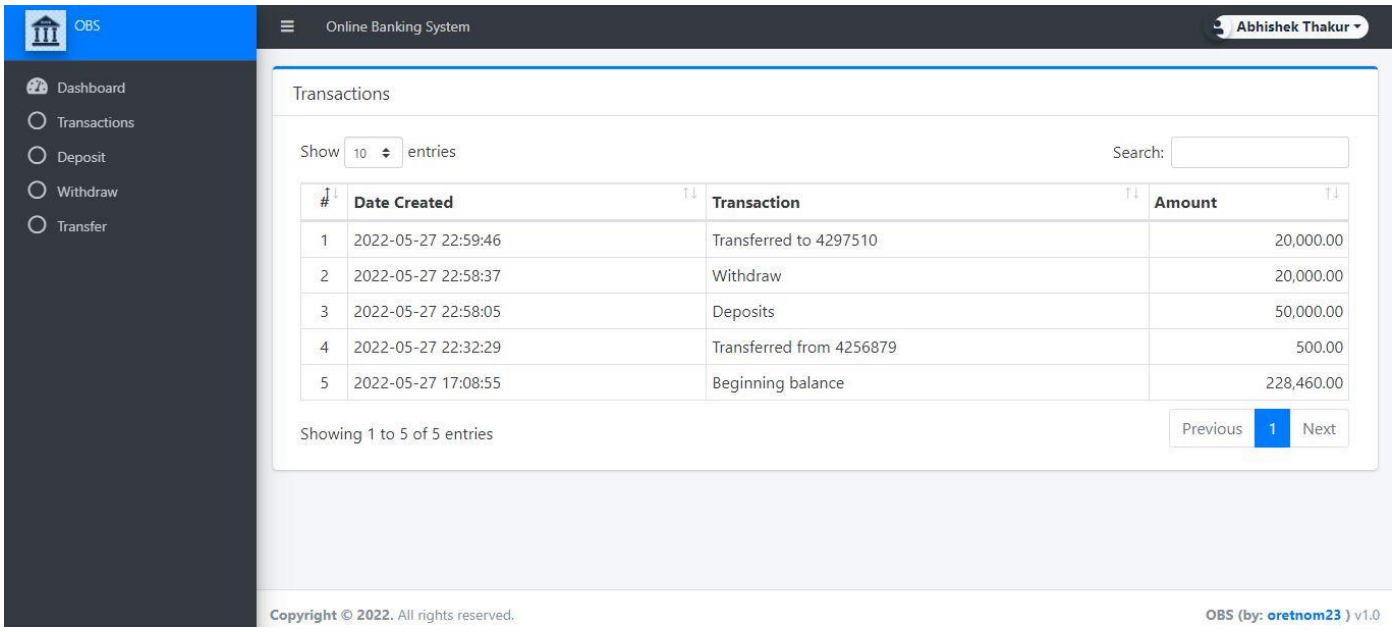


#	Account #	Name	Amount	Transaction	Date Created
1	4248634	Thakur, AbhishekSingh	20,000.00	Transferred to 4297510	2022-05-27 22:59:46
2	4297510	Sharma, AnjaliKumari	20,000.00	Transferred from 4248634	2022-05-27 22:59:46
3	4248634	Thakur, AbhishekSingh	20,000.00	Withdraw	2022-05-27 22:58:37
4	4248634	Thakur, AbhishekSingh	50,000.00	Deposits	2022-05-27 22:58:05
5	4256879	Mishra, VikashKumar	500.00	Transferred to 4248634	2022-05-27 22:32:29
6	4248634	Thakur, AbhishekSingh	500.00	Transferred from 4256879	2022-05-27 22:32:29
7	4256879	Mishra, VikashKumar	2,000.00	Withdraw	2022-05-27 22:30:56
8	4256879	Mishra, VikashKumar	200.00	Deposits	2022-05-27 22:30:13
9	4256879	Mishra, VikashKumar	5,000.00	Beginning balance	2022-05-27 22:28:24
10	4248634	Thakur, AbhishekSingh	228,460.00	Beginning balance	2022-05-27 17:08:55

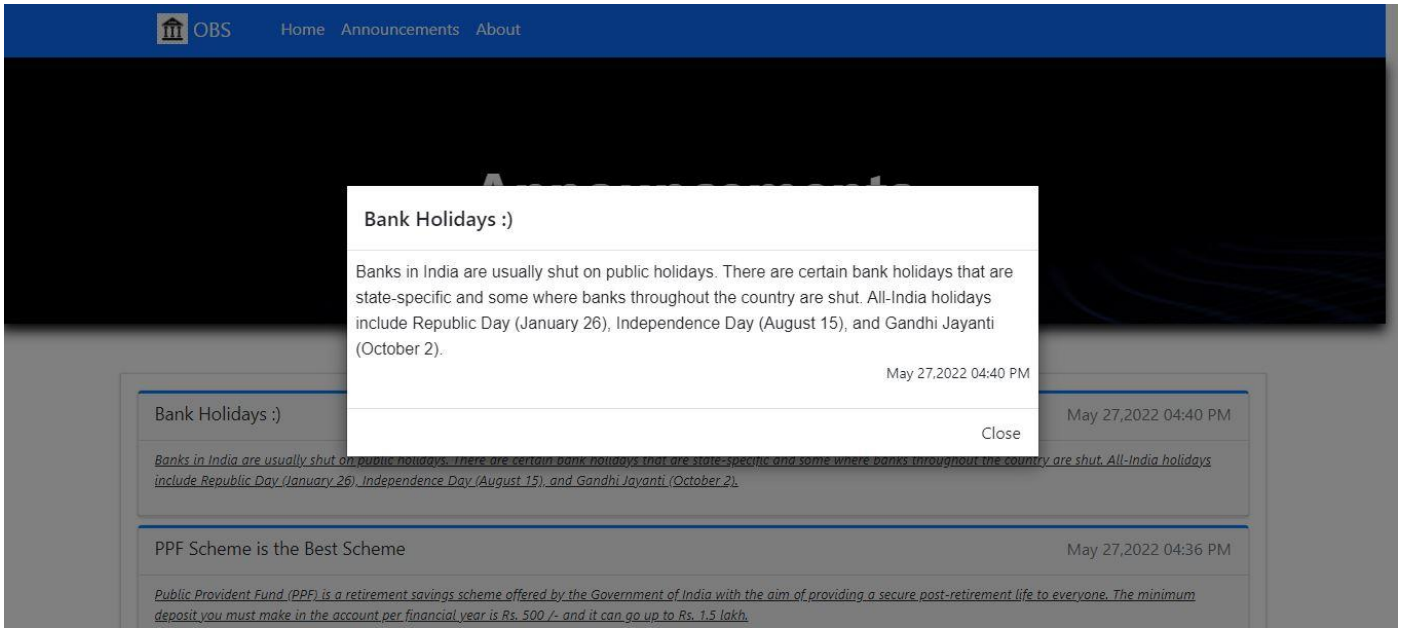
Client's Dashboard:



Client's Transaction List:



Public Announcement Page:



Coding

home.php

```php
<!-- Header-->
<header class="bg-dark py-5" id="main-header">
    <div class="container px-4 px-lg-5 my-5">
        <div class="text-center text-white">
            <h1 class="display-4 fw-bolder">Welcome <?php echo $_settings->info('name') ?></h1>
        </div>
    </div>
</header>
<!-- Section-->
<?php
$sched_arr = array();
$max = 0;
?>
<section class="py-5">
    <div class="container d-flex justify-content-center">
        <div class="card col-md-6 p-0">
            <div class="card-header">
                <div class="card-title text-center w-100">Login</div>
            </div>
            <div class="card-body">
                <form action="" id="login-client">
                    <div class="form-group">
                        <label for="email" class='control-label'>Email</label>
                        <input type="text" class="form-control" name="email" required>
                    </div>
                    <div class="form-group">
```

```php
                        <label for="password" class='control-label'>Password</label>
                        <input type="password" class="form-control" name="password" required>
                    </div>
                    <div class="form-group d-flex justify-content-end">
                        <button class="btn btn-sm btn-primary btn-flat">Login</button>
                    </div>
                </form>
            </div>
        </div>
    </div>
</section>
<script>
</script>
```

Admin Account Code:

admin > accounts > index.php

```php
<?php if($_settings->chk_flashdata('success')): ?>
<script>
    alert_toast("<?php echo $_settings->flashdata('success') ?>",'success')
</script>
<?php endif;?>
<div class="card card-outline card-primary">
    <div class="card-header">
        <h3 class="card-title">List of Accounts</h3>
        <div class="card-tools">
            <a href="?page=accounts/manage_account" class="btn btn-flat btn-primary">
        </div>
    </div>
    <div class="card-body">
        <div class="container-fluid">
            <table class="table table-bordered table-stripped" id="indi-list">
                <colgroup>
                    <col width="5%">
                    <col width="15%">
                    <col width="20%">
                    <col width="20%">
                    <col width="15%">
                    <col width="15%">
                    <col width="10%">
                </colgroup>
```

```php
            success:function(resp){
                if(typeof resp== 'object' && resp.status == 'success'){
                    location.reload();
                }else{
                    alert_toast("An error occured.",'error');
                    end_loader();
                }
            }
        })
    }
    $(function(){
        indiList = $('#indi-list').dataTable({
            columnDefs:[{
                targets:[6],
                orderable:false
            }],
        });
    })
</script>
```

Manage Account Code:

admin > accounts > manage_account.php

```php
<?php
if(isset($_GET['id']) && $_GET['id'] > 0){
    $qry = $conn->query("SELECT * from `accounts` where id = '{$_GET['id']}' ");
    if($qry->num_rows > 0){
        foreach($qry->fetch_assoc() as $k => $v){
            $$k=$v;
        }
}
}
?>
<div class="card card-outline card-primary">
    <div class="card-header">
    <h3 class="card-title"><?php echo isset($_GET['id']) ? 'Update Account' : "Create New Account"; ?></h3>
    </div>
    <div class="card-body">
        <div class="container-fluid">
            <form id="account-form">
                <input type="hidden" name="id" value='<?php echo isset($id)? $id : '' ?>'>
                <div class="form-group">
                    <label class="control-label">Account Number</label>
                    <input type="text" class="form-control col-sm-6" name="account_number" value="<?php echo
                </div>
                <hr>
```

Admin Transaction Deposit:

admin > transaction > deposit.php

```
<script>
    $(function(){
        $('#generate_pass').click(function(){
            var randomstring = Math.random().toString(36).slice(-8);
            $('[name="generated_password"]').val(randomstring)
        })
        $('[name="account_number"]').on('input',function(){
            if($('._checks').length > 0)
                $('._checks').remove()
            $('[name="account_id"]').val('')
            $('#name').val('')
            $('#balance').val('')
            $(this).removeClass('border-danger')
            $(this).removeClass('border-success')
            if($(this).val() == '')
            return false;
            $('button[form="account-form"]').attr('disabled',true)
            var checks = $('<small class="_checks">')
            checks.text("Checking availablity")
            $('[name="account_number"]').after(checks)
            $.ajax({
                url:_base_url_+'classes/Master.php?f=get_account',
                method:'POST',
                data:{account_number: $(this).val()},
                dataType:'json',
```

References

https://www.w3schools.com/php/
https://www.Bootstrap.com/
https://www.quora.com/What-are-the-requirements-for-a-online-bank-management-system