BANKING MANANAGEMENT SYSTEM

Based on Force.com

Table of Contents

Northern India Engineering College

Department of Computer Science Engineering

Certificate

This is to certify that Naman Mittal, a student of CSE-T1 has successfully completed the research on the below mentioned project under the guidance of Mr. Mohit Sharma during the Summer Training Program.

Declaration

I hereby declare that this Project Report titled "Banking Management Project" submitted to the Department Of Computer Science Engineering, Northern India Engineering College (GGSIPU) is a record of original work done by me under the guidance of Mr. Mohit Sharma.

The information and data given in this report is authentic to the bestof my knowledge.

This Project Report is not submitted to any other university or institution for award of any degree, diploma or fellowship or published any time before.

Project: Banking Management System

1.1 Introduction

A cloud based management system is designed to handle all banking functionalities like the primary information required to calculate monthly statements of customer account which include monthly statement of any month, performing transactions on accounts through various methods such as cheque, card or ATM, approval of loan, real-time balance check and e statement through net banking was implemented using a cloud based service (platform as a service) Force.com. The system also keeps a record of services issued to customers like cheque books, cards or demand draft etc. also keeps s record of employees of the bank and their details.

This project intends to introduce more user friendliness in the various activities such as record updating, 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 updating can also be accomplished by using the account number with all the details being automatically generated.

Migrating to a cloud based platform – Force.com, the hassle of managing physical servers, complex backend development, worries of security can be left with the Force.com's services, this reduces the development effort and time of the IT Dept. of the Bank, also helps save a lot of financial resources spent in management of physical servers (H/W Cost), their hardware and sensitive backend software.

1.2 Objective

The objective of this project was to migrate the bank's management system to a Cloud based platform Force.com which is a PaaS. Force.com manages the backend database, Physical storage and server hardware of the system. The hassle of managing physical servers, complex backend

development, worries of security can be left with the Force.com's services, this reduces the development effort and time required to automate bank's procedures such as management of primary information required to calculate monthly statements of customer account which include monthly statement of any month, performing transactions on accounts through various methods such as cheque, card or atm, approval of loan, real-time balance check and e statement through net banking, etc. Moving to Force.com eliminates the need to buy and setup physical hardware for servers and robust computers to run the system hence reducing the operational cost of bank's management system drastically. It also simplifies and eases the software development process facilitating the developers to build robust and fine systems quickly and easily.

Another objective was to introduce more user friendliness in the various activities such as record updating, 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 updating can also be accomplished by using the account number with all the details being automatically generated.

The main objective of this project is providing the different types of customer facility as well as reducing the effort required by IT team of the bank in implementing those facilities.

- It should fulfill almost all the process requirements of any Bank.
- It should increase the productivity of bank by utilizing the working hours more and more, with minimum manpower.

System Requirement Specification

2.1 Functional Requirements

- System should keep a record of all the established branches of the bank and details of the branches like address, Contact information, etc.
- System should be able to keep a record of Bank's employees working in different branches and details of employees like their Name, Contact Info, Address, in which branch they are posted, their respective positions and salaries etc.
- System should generate unique Employee ID automatically when a new Employee record is created.
- System should be able to keep a record of Bank's customers and their details like Name, Address, DOB, Aadhar number, Contact info etc., and can assign a relationship manager to them for managing their requirements/ Requests.
- System should generate unique Customer ID automatically when a Customer record is created.
- Bank Staff should be able create new account and assign the accounts to customers present in the database (done in previous step). One customer can have many accounts of different type like a customer having a current account can have also have a Fixed Deposit Account. ROI will be automatically selected upon creation of Account depending upon the type of account.
- Account number should be a unique number automatically generated while creating an account.
- System should allow transactions to be made from different transaction methods both offline and online and assign an automatically generated unique Transaction ID for easy tracking.
- Bank staff should be able to perform offline transactions on accounts if customer uses any offline transaction methods like Cheques and Demand Drafts.
- Online transactions made by the customer through various transaction method like ATM, Card or Online Transfer should be immediately reflected to the central database on his account.
- Bank should be able to manage records of issued Cheque Books to an Account Holder as well as create records to issue new Cheque Books.

- Bank Should be able to keep a record of Loans sanctioned and send new loan applications after initial verification to Manager for final amount, ROI, Payback Date, EMI etc. and ass9igna unique Load ID to every sanctioned loan.
- Loan must be sent for approval to the manager before it is granted and taken into database for processing.
- Bank should be able to manage the records of cards issued to customers and their details like Card No. , CVV No. , Pin no. , etc. card no. and CVV should be uniquely and automatically generated by the system.
- Bank should be able to manage the records of Demand Drafts issued to customers and their details like payee amount etc. System should assign a unique ID to it.
- Users should be able to Transaction History(Transaction ID, Amount and Type) and of their Accounts
- Users should be able to make online transactions via Bank's Online Fund Transfer Service, authenticated by a unified Salesforce.com ID and a password that works for all online services.
- Users should be able to make online transactions via cards on different websites, authenticated by a unified ID and a password that works for all online services.
- Users should be able to withdraw / Deposit Cash in their accounts via ATM Service.

2.2 Non - Functional Requirements

- System should be fast to reflect the change in account balance after transaction is made through any method or branch , it should be immediately reflected to central database
- User Interface of the Bank's End should be light and basic to reduce the bandwidth consumption.
- User interface of Customer's Services end should be simple, attractive, reliable and fluid.
- In case of failed transaction system should roll back any changes to the account balance.

- Multiple accounts from same customer should be automatically linked.

2.3 Non – Requirements (Constraints)

- Users should not be able to see the details of any account other than their own through any screen available in their services.
- Rights and roles of accounts of staff should be clearly defined no employee should be able to access features and functionalities that is beyond their rights.
- System should not accept invalid data like incorrect account no. or card no. etc.
- System should not apply the transaction if any of the input criteria mismatches the records.
- System should not allow users or staff members to make transactions of amount more than the account balance as well as more than the limit specified in the user's services.

Technology and Tools

3.1 A Brief Introduction to Force.com Platform

Force.com lets developers rapidly create and deploy trusted cloud applications that are rock solid, secure, and scalable without having to worry about provisioning hardware or application stacks. To help you go faster, Force.com delivers out-of-the-box tools and services to automate your business processes, integrate with external applications, deliver mobile experiences and more. Read on to learn why Force.com is the #1 Application platform as a service.

Force.com is the only platform that lets you build powerful enterprise applications without writing a line of code. You can create apps by dragging and dropping components into the Lightning App Builder.

Automate business processes with the Lightning Process Builder and modify your data object relationships with the Schema Builder.

Force.com includes a powerful, tightly integrated cloud database with features that makes it faster and easier to create applications. Add fields to standard objects or create custom objects to store information that's unique to your organization - all with a point and click interface. Using Lightning Connect you can connect to external data sources and use them seamlessly in your application with no need to sync or migrate your data.

While you can do a lot with Lightning, some apps require more customization. When you need to go further, you can write code that executes on the Force.com platform with Apex, a strongly-typed, object-oriented programming language. With Apex code, you can build powerful business logic, create custom triggers and even perform asynchronous calls that run in a separate thread of execution.

Force.com includes robust, out of the box security controls that handle user authentication and let you specify which users or groups of users can view, create, edit, or delete any record or field in the app. Using profiles, sharing rules, permission sets, and role hierarchies you get precision control over how data is shared across your users.

You can easily customize page layouts, specify filter criteria for list views and create tabs to access custom objects - all without code! If you need to extend the Salesforce UI you can create re-usable Lightning Components using HTML, CSS and JavaScript or build entirely new user interfaces with Visual force and your JavaScript framework of choice.

Technology	Description
Multitenant architecture	An application model in which all users and apps share a single, common infrastructure and code base.
Metadata-driven development model	An app development model that allows apps to be defined as declarative "blueprints," with no code required. Data models, objects, forms, workflows, and more are defined by metadata.
API Access	Several application programming interfaces (APIs) provide direct access to all data stored in Force.com from virtually any programming language and platform. • The SOAP API and REST API integrate your organization's data with other applications • The RESTful Bulk API (also available using Data Loader) loads or deletes large numbers of records • The Metadata API manages customizations in your organization (also available using the Force.com Migration Tool) • The Chatter REST API accesses Chatter feeds and social data • The Streaming API provides notifications reflecting data changes in your organization
Apex	The world's first on-demand programming language, which runs in the cloud on the Force.com platform servers.
Visualforce	A framework for creating feature-rich user interfaces for apps in the cloud.
Mobile Access	With Salesforce mobile apps, you can access custom apps built using the Force.com platform's point-and-click development tools. Your users can access those apps on their mobile devices–and you don't have to learn any mobile programming languages.
AppExchange directory	A Web directory where hundreds of Force.com apps are available to Salesforce customers to review, demo, comment upon, and/or install. Developers can submit their apps for listing on the AppExchange directory if they want to share them with the community.

Figure: The Technologies behind a Force.com Platform App

3.2 Force.com App Builder

Objects, Fields and Relationships

Applications in force.com are built as objects interrelated to each other through various relations. An **Object** is basically a table if talk in RDBMS language, it represents a table Force.com platform objects not only provide a structure for storing data but they also power the interface elements that allow users to interact with the data, such as tabs, the layout of fields on a page, and lists of related records. Because any object can correspond to a tab, and an ordered collection of tabs makes up an app, objects make up the heart of any app that we create with the platform and **Fields** represent the columns in that table. Every object has a standard field provided by default when object is created. We can add as many custom fields as we want.

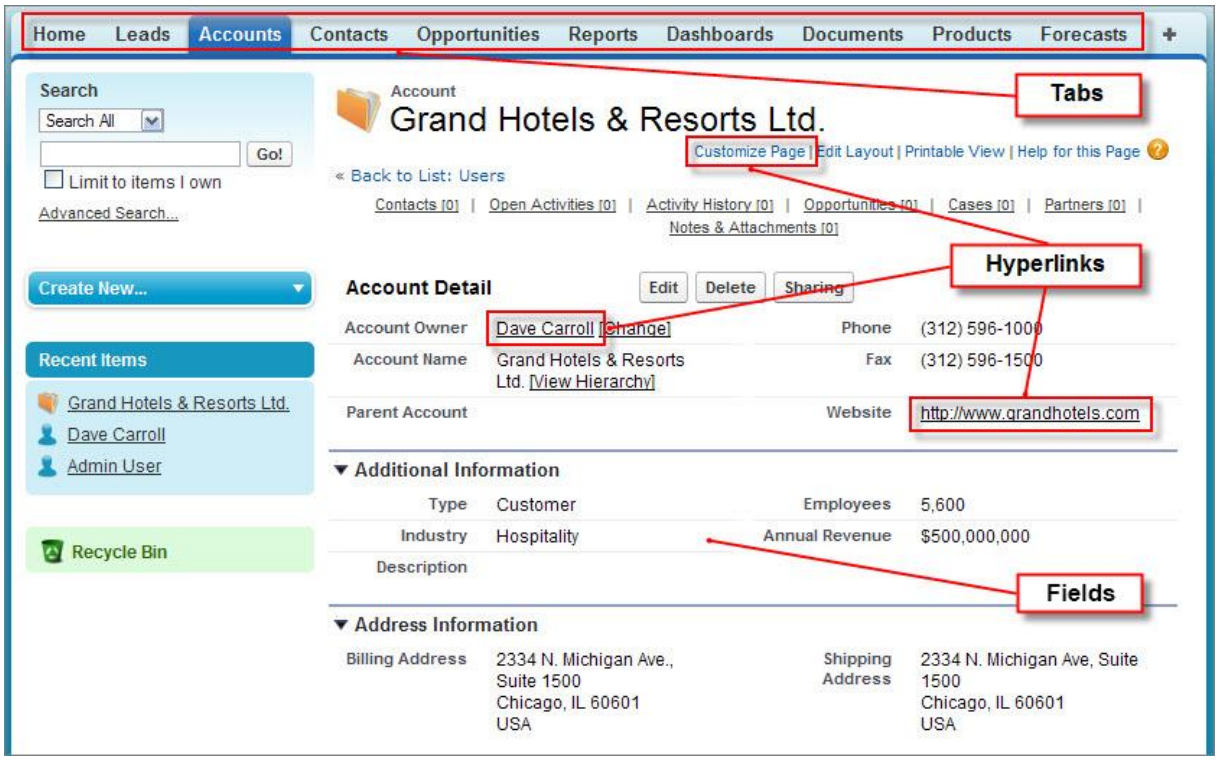


Figure: The Basics of an App's User Interface

Tabs

Across the top of the app is a set of tabs that segment the app into different parts. Each tab corresponds to a type of object, such as an account or contact, and within a tab you can perform actions on particular records of that tab's type. For example, when you click on the Accounts tab, you can create a new record for the “Acme” account. You can also edit existing accounts, or use a list view to filter lists of accounts by certain criteria. Most app development work revolves around creating tabs and defining the data and behaviours that support them.

Fields

Displayed within each record is a selection of fields, which is how the Force.com platform houses and organizes information. For example, a contact record includes fields such as Last Name, Home Phone, Mailing City, Title, Birthdate, Reports To, and Account. When developing a new app, you can customize which fields appear for a given type of record—such as for contact records—as well as how they are organized. In a Force.com platform app, users enter information with writable fields on an edit page and view that information with read-only fields on a detail page.

Links

Finally, because Force.com platform apps are delivered in a Web browser, they use links to provide navigation to related data. For example, on an account detail page, there are links to related records, such as the contacts who belong to the account and the sales user who manages the account. Other links take you to recently visited records and to areas of the app where users can set personal preferences. Links provide navigation within an app and to external Web sites.

Validation Rules for Data Validation

Validation rules verify that the data a user enters in your app meets the standards that you specify. If it doesn't, the validation rule prevents the record from being saved, and the user sees an error message that you define either next to the problematic field or at the top of the edit page.

Page Layouts

Page layouts are powerful tools for creating a good experience for our users, but it's crucial that we remember one important rule: page layouts should never be used to restrict access to sensitive data that a user shouldn't view or edit. Although we can hide a field from a page layout, users can still access that field through other parts of the app, such as in reports or via the API.

Relationships

Master-detail

Closely links objects together such that the master record controls certain behaviours of the detail and sub detail record. For example, you can define a two-object master-detail relationship, such as Account—Expense Report, that extends the relationship to sub detail records, such as Account—Expense Report—Expense Line

Item. You can then perform operations across the master—detail—sub detail relationship.

Behaviours of master-detail relationships:

Deleting a detail record moves it to the Recycle Bin and leaves the master record intact; deleting a master record also deletes related detail and sub detail records. Undeleting a detail record restores it, and undeleting a master record also undeletes related detail and sub detail records. However, if you delete a detail record and later, separately, delete its master record, you cannot undelete the detail record, as it no longer has a master record to relate to.

By default, records can't be represented in master-detail relationships. Administrators can, however, allow child records in master-detail relationships on custom objects to be represented to different parent records by selecting the Allow representing option in the master-detail relationship definition.

The Owner field on the detail and sub detail records is not available and is automatically set to the owner of the master record. Custom objects on the "detail" side of a master-detail relationship can't have sharing rules, manual sharing, or queues, as these require the Owner field.

Detail and sub detail records inherit security settings and permissions from the master record. You can't set permissions on the detail record independently.

The master-detail relationship field (which is the field linking the objects) is required on the page layout of the detail and sub detail records.

Lookup

Links two objects together. Lookup relationships are similar to master-detail relationships, except they do not support sharing or roll-up summary fields. With a lookup relationship, you can:

Link two different objects.

Link an object with itself (with the exception of the user object; see Hierarchical). For example, you might want to link a custom object called “Bug” with itself to show how two different bugs are related to the same problem.

Workflow Rules

Workflow rules can be broken into two main components.

Criteria: the “if” part of the “if/then” statement. In other words, what must be true of the record for the workflow rule to execute the associated actions?

Actions: the “then” part of the “if/then” statement. In other words, what to do when the record meets the criteria.

In the raining example, the criteria is “it’s raining” and the action is “bring an umbrella”. If the criteria isn’t met (it isn’t raining), then the action isn’t executed (you don’t bring an umbrella).

Approval Process

Approvals take automation one step further, letting you specify a sequence of steps that are required to approve a record. An approval process automates how records are approved in Salesforce. An approval process specifies each step of approval, including who to request approval from and what to do at each point of the process.

If all approval requests are approved, change the status to Approved and unlock the record. If any approval requests are rejected, change the status to Rejected and unlock the record.

Object-Level Security

The bluntest way that we can control data is by preventing a user from seeing, creating, editing, or deleting any instance of a

particular type of object, like a position or review. Object-level access allows us to hide whole tabs and objects from particular users, so they don't even know that type of data exists.

On the platform, we set object-level access with object permissions in user profiles and permission sets. We'll learn more about them in a little bit.

Field-Level Security

A variation on object-level access is field-level access, in which a user can be prevented from seeing, editing, or deleting the value for a particular field on an object. Field-level access allows us to hide sensitive information like the maximum salary for a position or a candidate's social security number without having to hide the whole object.

On the platform, we set field-level access with field permissions, also in profiles and permission sets. We'll also learn more about them shortly.

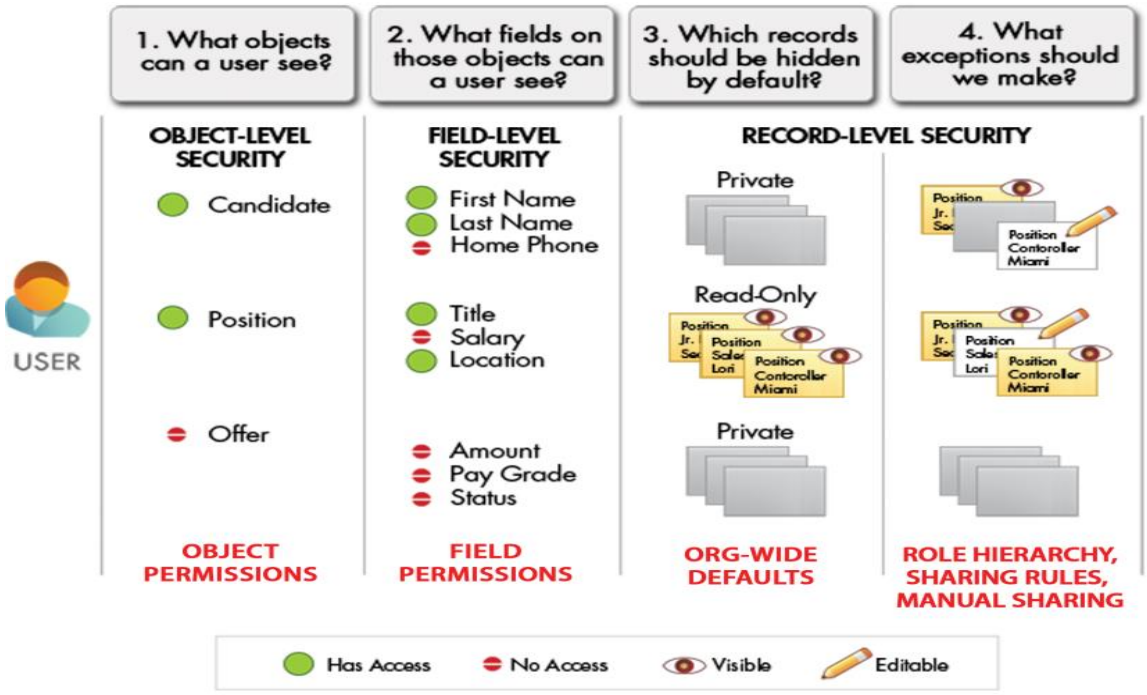
Record-Level Security

To control data with a little more finesse, we can allow particular users to view an object, but then restrict the individual object records that they're allowed to see. For example, record-level access allows an interviewer like Melissa Lee to see and edit her own reviews, without exposing the reviews of everyone else on her team.

On the platform, we actually have several ways of setting record-level access rules:

- *Organization-wide defaults* allow us to specify the baseline level of access that a user has in your organization. For example, we can make it so that any user can see any record of a particular object to which their object permissions give them access, but so that they'll need extra permissions to actually edit one.
- *Role hierarchies* allow us to make sure that a manager will always have access to the same records as his or her subordinates.

- *Sharing rules* allow us to make automatic exceptions to organization-wide defaults for particular groups of users.
- *Manual sharing* allows record owners to give read and edit permissions to folks who might not have access to the record any other way.



Apex Triggers

Triggers is an Apex code that Executes before or after the following types of DML Operations:

- Insert
- Update

- Delete
- Merge
- Upsert
- Undelete

Triggers are Divided into 2 Types:-

- Before Triggers
- After Triggers

1. **Before Trigger: -** Before Trigger can be Used to Update or Validate values of Record before they are saved to the database.
2. **After Trigger: -** After Trigger can be Used to Access field values of the Record that are stored in Database and use this values to make changes in other Records.

   Syntax:

   Trigger ***triggerName*** on ObjectName ***(trigger_event)***

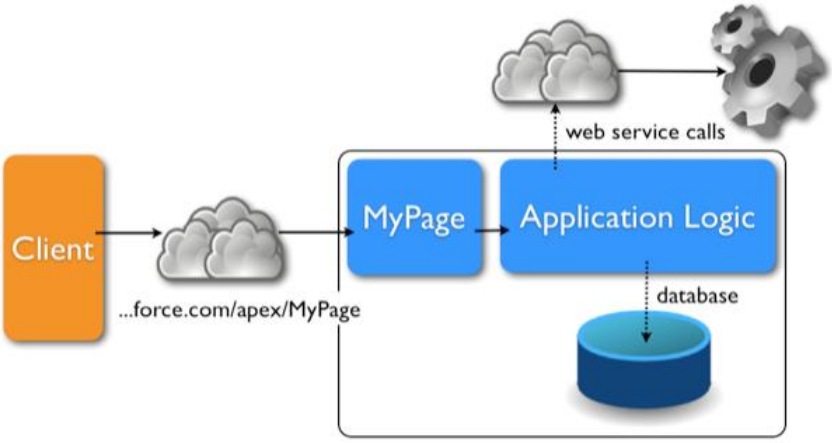
   {

   //Code block;

   }

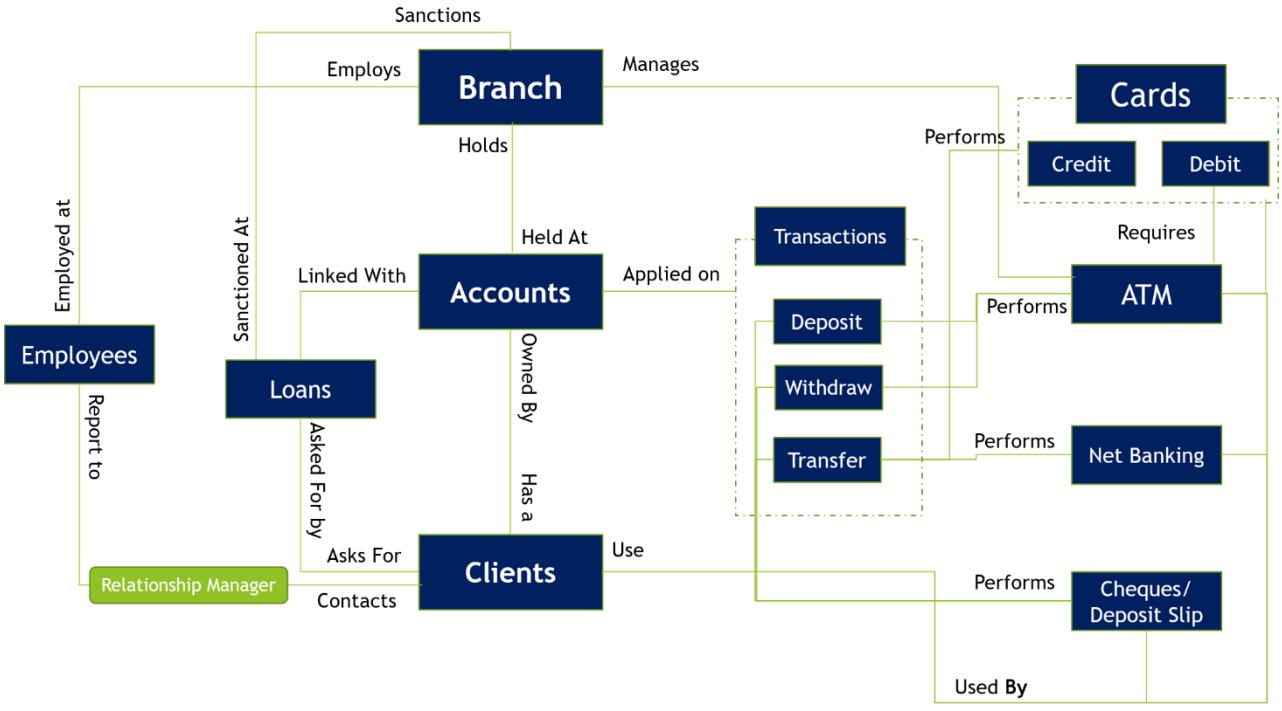
Visual Force

A developer creates Visual force pages by composing components, HTML, and optional styling elements on the Force.com platform. Just like HTML, Visual force can integrate with any standard web technology or JavaScript framework to allow for a more animated and rich user interface. Each page is then accessible by a unique URL. When someone accesses a page, the server renders the page.

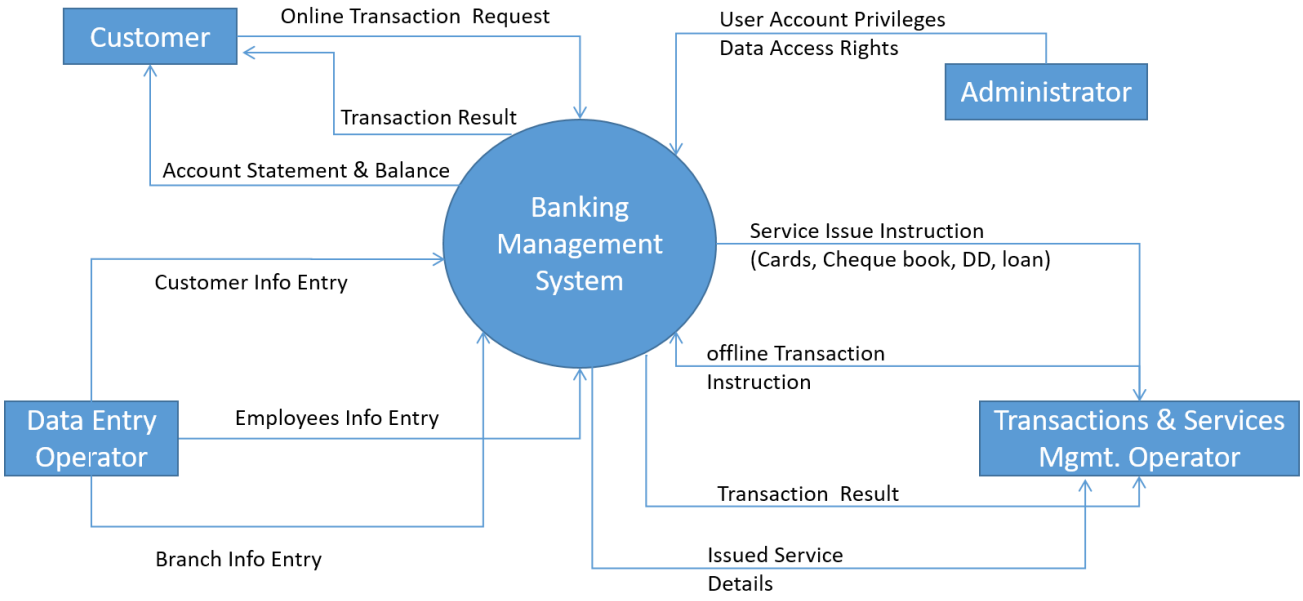


System Design

4.1 System Architecture



4.2 DFD



4.3 Implementation on Force.com Platform

Objects and Details

Object Name: Branch

***Standard Field*: -**

Branch Name (Text)
Owner (Lookup (User, Queue))

Custom Fields: -

Address Line 1 [Text]
Address Line 2 [Text]
Address Line 3 [Text]
City [Text]

State [Text]
Contact Number [Phone]
Landmark [Text]
Pin Code [Number]

Validation Rules:-

1) **Address Verification**

You must give an address line

Error Condition:-

ISBLANK (Address_Line_1__c)

2) Pin Code Verification

You must enter a valid postal code

Error Condition:-

LEN (TEXT (Pin_Code__c)) <> 6

Object Name: Employee

Standard Field: -

Employee ID (Auto-Number)

***Custom Fields* : -**

Address Line 1 [Text]
Address Line 2 [Text]
Address Line 3 [Text]
Branch [Master-Detail(Branch)]
City [Text]
Contact Number [Phone]
Email-ID [Email]
First Name [Text]
Last Name [Text]
Middle Name [Text]
Pin Code [Number]
Position [Pick list]
Salary [Currency]
State [Text]

Relationships:-

Master –> Detail (Branch -> Employee**)**

Validation Rules:-

1) Pin Code Verification

you must enter a valid postal code.

Error Condition:-

LEN (TEXT (Pin_Code__c)) <> 6

Object Name: Customer

***Standard Field*: -**

Customer ID (Auto–Number: Standard)

Custom Fields: -

First Name [Text]

Last Name [Text]

Mobile Number [Phone]

Date of Birth [Date]

Email ID [Email (unique)]

Address Line 1 [Text]

Address Line 2 [Text]

Address Line 3 [Text]

Aadhar Number [Number]

City [Text]

State [Text]
Pin Code [Number]

Relationship Manager [Lookup (Employee)]

Relationships:-

Master –> Detail (Customer -> Account)

Lookup (Employee -> Customer**)**

Validation Rules:-

1) **Pin Code**

   Pin code must be 6 digits

   Error Condition:-

   If (Len (TEXT (Pin_Code__c)) = 6, false, true)

Object Name: Bank Account

***Standard Field*: -**

Account Number (Auto-Number)

Custom Fields: -

Account Balance [Currency]
Account Holder [Master-Detail (Customer)]
Account Type [Pick list]
Interest per Day [Formula (Currency)]
Opening Date [Date]
Rate of Interest [Pick list]

Relationships:-

Master – Detail (Customer -> Bank Account**)**

Validation Rules:-

1) **Minimum Amount**

   The account should have at least $1000

   Error Condition:-

   Account_Balance__c <= 1000

Object Name: Cheque Book

Standard Field: -

MICR (Auto – Number)

Custom Fields: -

Account Number [Lookup (Bank Account)]
Issued To Name [Formula (Text)]
Number to Leaves [Number]

Relationships:-

Lookup (Bank Account -> Cheque Book**)**

Object Name: Transaction

Standard Field: -

Transaction ID (Auto-Number)

Custom Fields: -

Amount [Currency]
Method [Pick list]

Source Account [Master-Detail (Bank Account)]
Type [Pick list]

Relationships:-

Lookup (Bank Account -> Transaction)

Object Name: Loan
Standard Field: -

Loan ID (Auto-Number)
***Custom Fields* : -**
EMI Option [Pick list]
Instalment Due Date [Date]
Loan Amount [Currency]
Loan Type [Pick list]
Payback Due Date [Date]
Rate of interest [Number]
Sanction Date [Date]
Total Amount [Formula(currency)]
Total Interest [Formula(currency)]

Object Name: Card

Standard Field: -

Card Number (Auto-Number)

Custom Fields: -
Bank Account [Master-Detail (Bank Account)]

CVV [Number (Unique)]
Expiry Date [Date]
Pin [Number]
Type [Pick list]
Subtype [Pick list]

Relationships:-

Master – Detail (Bank Account -> Card)

Object Name: Card User

Standard Field: -

Card User Name (Text)

Custom Fields: -

Card Number [Master-Detail (Card)]
CVV [Number]
Expiry Date [Date]
Withdraw Amount [Currency]

Relationships:-

Master –> Detail (Card -> Card user)

Validation Rules

1) **User Verification**
   The values entered do not match.
   Error Condition:-

```
IF( Expiry_Date__c = Card_Number__r.Expiry_Date__c , IF( CVV__c =
Card_Number__r.CVV__c , IF( Name =
Card_Number__r.Bank_Account__r.Account_Holder__c ,TRUE ,FALSE)
,FALSE),FALSE)
```

Apex Trigger

```
Trigger ins1 on Card_User__c (after insert) {
 list<Card_User__C> cu = trigger.new;
   list<Transaction__c> tlist = new list<Transaction__c> ();
   Transaction__c t = new Transaction__c
(Method__c = 'Debit Card', Type__c = 'Withdrawal');
   for (Card_User__c c: cu) {
   t.Linked_Account__c = c.Card_Number__r.Bank_Account__r.Name;
```

```
        t.Amount__c = c.Withdraw_Amount__c;
       tlist.add (t);
    }
    insert tlist;
}
```

This trigger inserts a record in transaction object when a transaction is made through card, the workflow rule applied on transaction object then updates the account balance field in Bank Account object.

Object Name: ATM USER

Standard Field: -

ATM User Name (Text)

Custom Fields: -

Card Number [Master-Detail (Card)]
Pin [Number]
Withdraw Amount [Currency]

Relationships:-

Master – Detail (Card -> ATM User)

Validation Rules:-

1) **Pin Verification**

   The PIN entered should be correct

   Error Condition:-

   Card_Number__r.PIN__c <> PIN__c

Apex Trigger

```
Trigger ins2 on ATM_User__c (before insert) {
    list<ATM_User__C> au = trigger.new;
    list<Transaction__c> tlist = new list<Transaction__c> ();
    Transaction__c t = new Transaction__c
(Method__c = 'Debit Card', Type__c = 'Withdrawal');
    for (ATM_User__c a: au) {
        t.Linked_Account__c = a.Card_Number__r.Bank_Account__r.Name;
        t.Amount__c = a.Withdraw_Amount__c;
        tlist.add (t);
    }
```

```
    insert tlist;
}
```

This trigger inserts a record in Transaction Object after a record is inserted in this object when a transaction is occurred through ATM. The workflow rule applied on transaction object then updates the account balance field in Bank Account object.

Object Name: Online Transfer

Standard Field: -

Service Name (Text)

Custom Fields: -

Amount [Currency]
Destination Account [Lookup (Bank Account)]
Source Account [Lookup (Bank Account)]

Relationships:-

Lookup (Bank Account -> Online Transfer)

Apex Trigger

```
trigger ins3 on Online_Transfer__c (before insert) {
  list<Online_Transfer__c> ou = trigger.new;
    list<Transaction__c> tlist = new list<Transaction__c>();
    Transaction__c tw = new Transaction__c(Method__c = 'Online Transfer', Type__c = 'With
drawal');
    Transaction__c td = new Transaction__c(Method__c = 'Online Transfer', Type__c = 'Depo
sit');
    for(Online_Transfer__c o:ou){
      tw.Linked_Account__c = 'a012800000Z0ql9';
        tw.Amount__c = o.Ammount__c;
        td.Linked_Account__c = 'a012800000Z0qld';
        td.Amount__c = o.Ammount__c;
        tlist.add(tw);
        tlist.add(td);
    }
    insert tlist;
}
```

This trigger inserts a record in transaction object when a transaction is occurred through online transfer. The workflow rule applied on transaction object then updates the account balance field in Bank Account object.

Object Name: Demand Draft

Standard Fields:-

Draft Number (Auto-Number)
Owner (Lookup (User, Queue)
Custom Fields:-
Amount [Currency]
Payee [Text]

Workflow Rules:-

Two workflow rules operate on transaction object

1.) Update(rule name)
    This rule updates the Account Balance field of Bank Account object when a withdrawal transaction is inserted in transaction object. It updates the account balance after deducting the withdrawn amount.
2.) Update 2 (rule name)
    This rule updates the Account Balance field of Bank Account object when a Deposit transaction is inserted in transaction object. It updates the account balance after crediting the deposited amount.

Approval Process

Approval process is used to get the approval for the loan, when a loan record is inserted the record is sent for approval to the CEO, before approval changes can be made using recall action, once seen for approval record is locked for any change and then final approval action takes place for acceptance or rejection of loan by CEO/Manager

User Interface & Screenshots

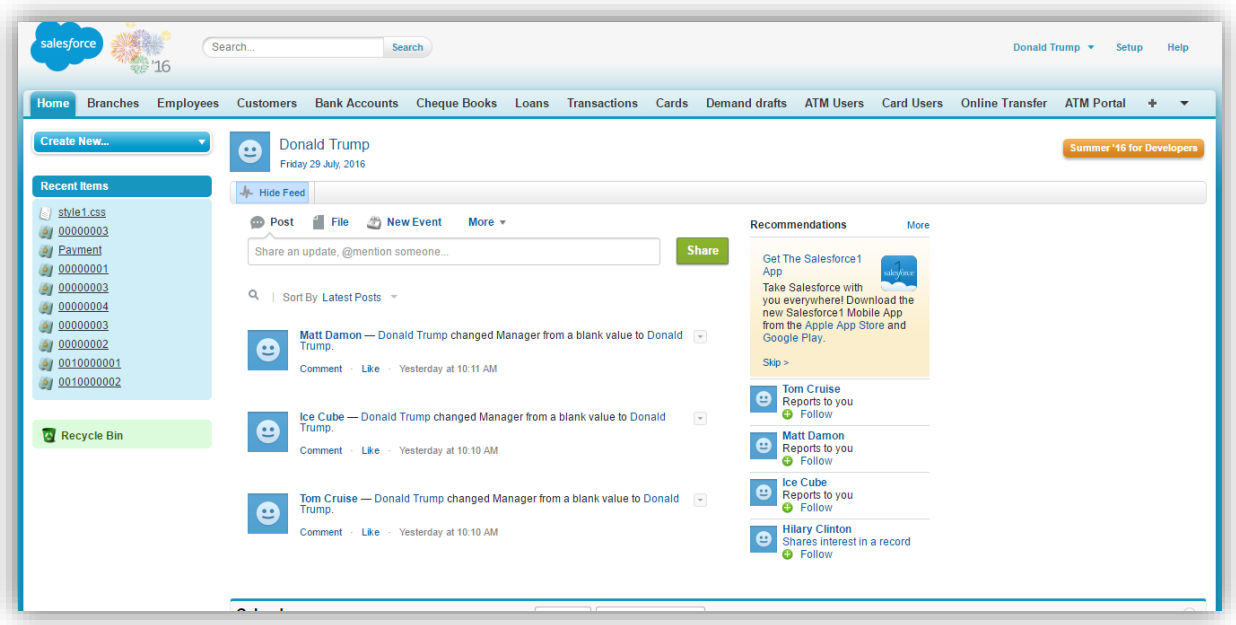
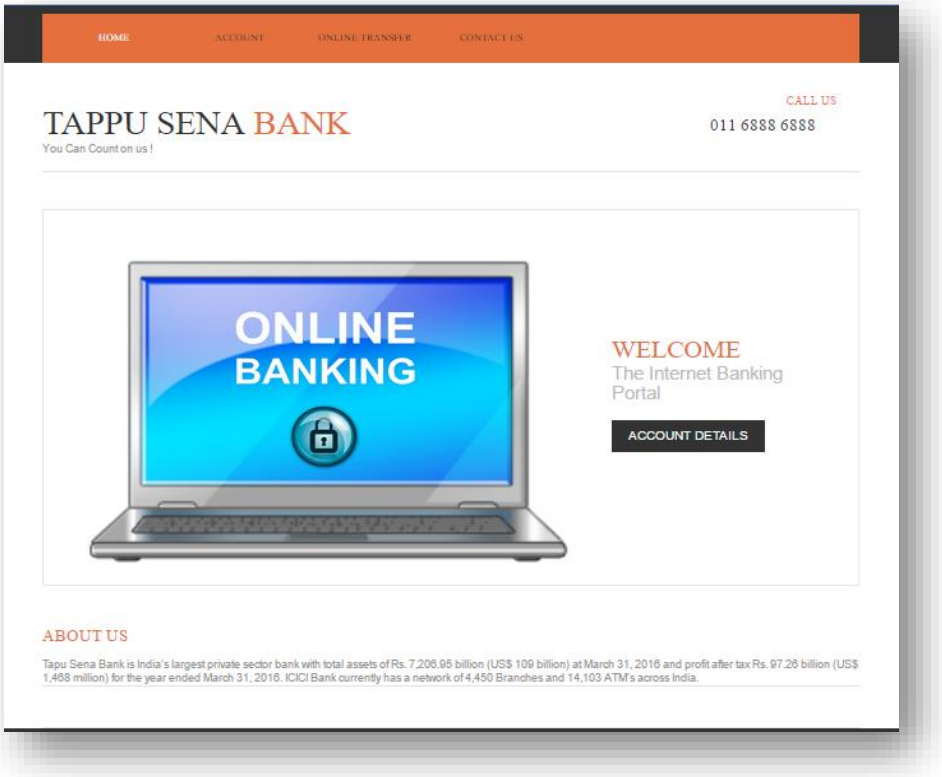
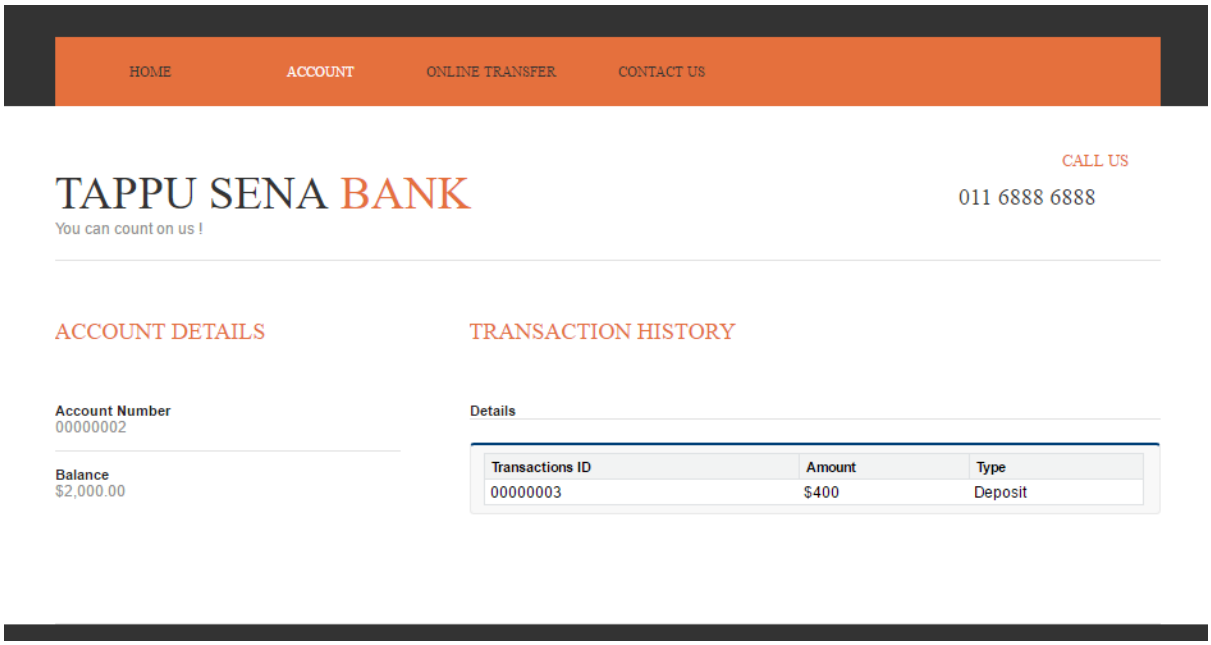
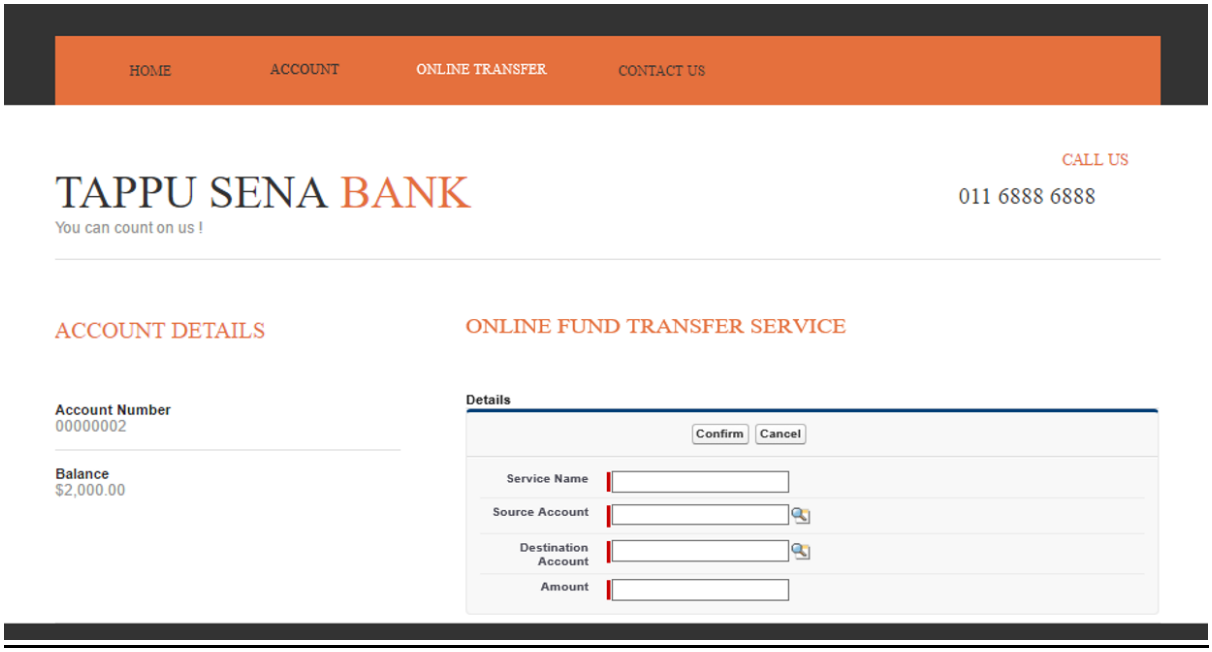


Figure: Banks User Interface

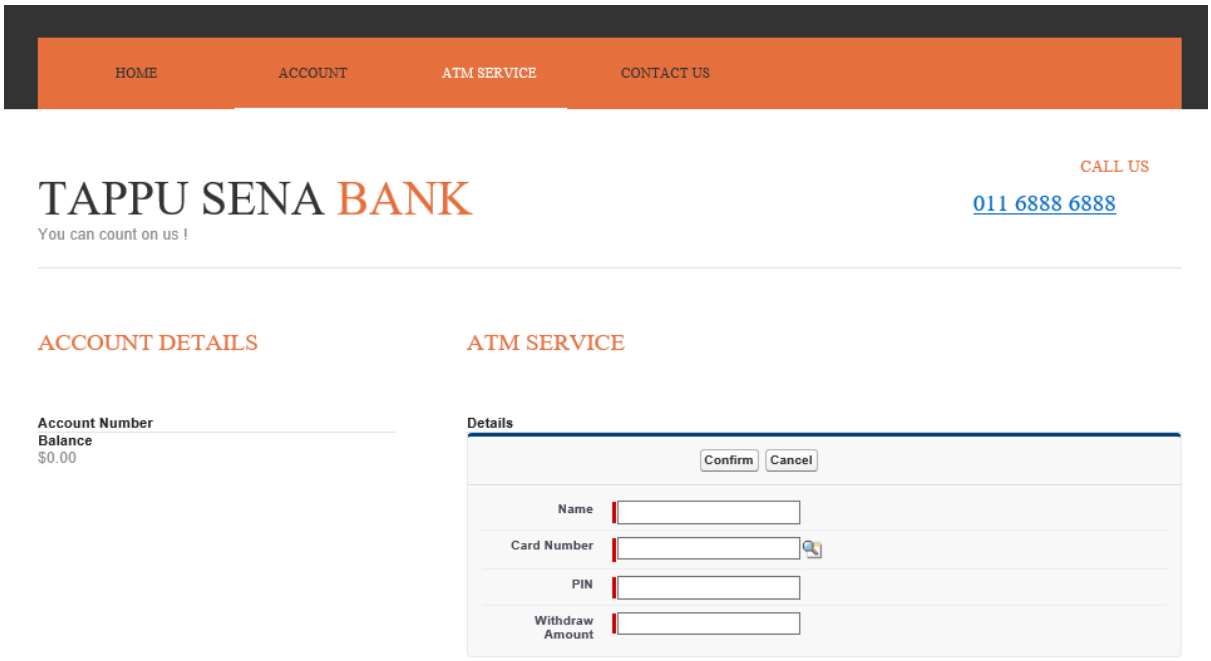
>> Online Transfer Portal



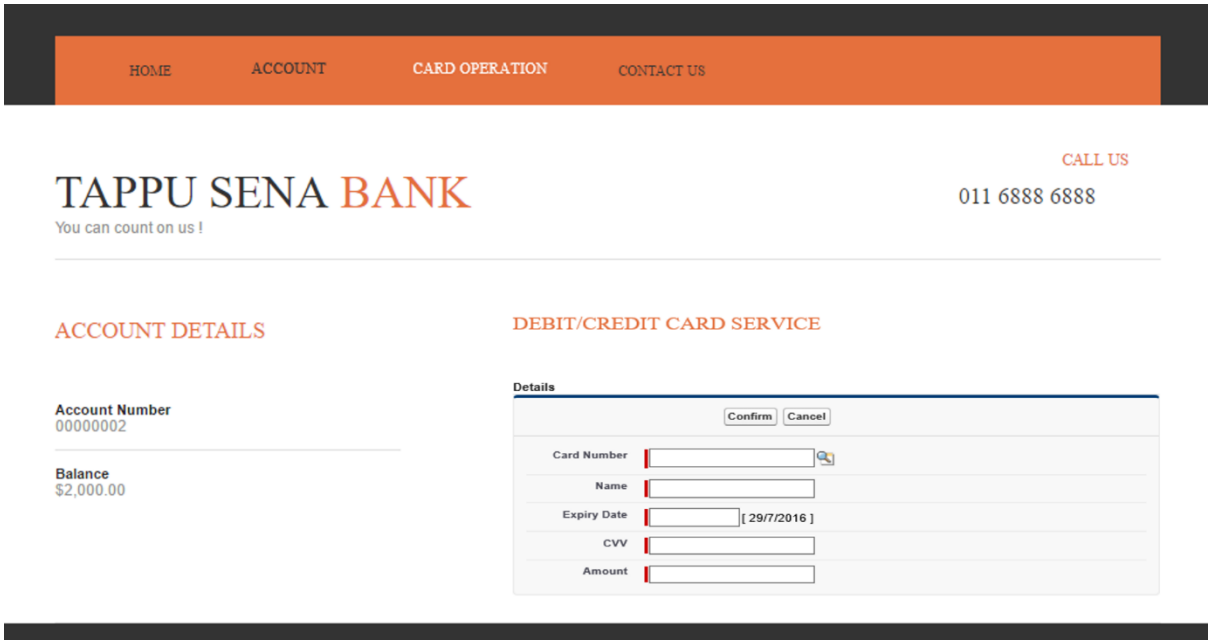


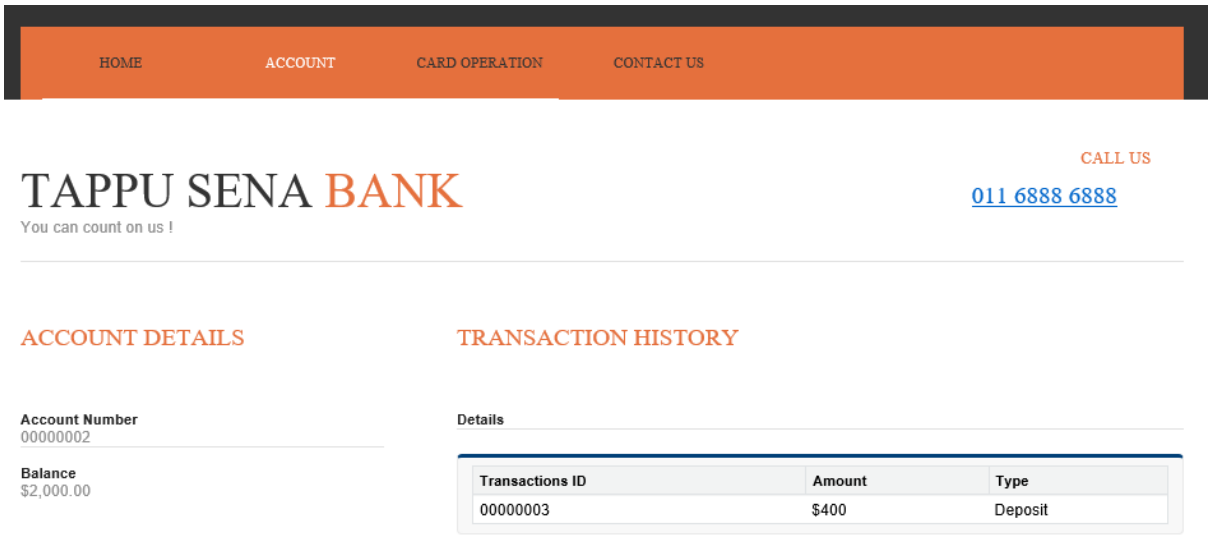


>> ATM User Interface



>> Card Payment Portal Page





The GUI has been completely customised using visual force pages, HTML 5 and CSS5, Please Refer the Live Project for more details on source code and functionality.

References

- https://trailhead.salesforce.com/

- https://developer.salesforce.com/forums/