**Borrow Loan Company**

**Project Report**

**Abstract**

Borrow loan company is a Responsive and user-friendly web application for financial services aimed at simplifying the loan processing for both customers and financial institutions. The application offers a user-friendly interface that allows customers to easily apply for various types of loans, credit cards, stocks listing and account management. Key features include a comprehensive loan calculator, Blogs, Multi-Device Access, easy Loan Applications and secure document uploads.

Contents

1. **Introduction**
   1. Summery ...................................................................................................................... 4
   2. Existing System ............................................................................................................ 4
   3. Proposed System ........................................................................................................... 4
2. **Technology**
   * 1. Python .................................................................................................................. 7
     2. HTML .................................................................................................................. 8
     3. MYSQL ................................................................................................................ 8
3. **System Configuration**

3.1.1 Hardware Configuration .............................................................................................. 9

3.1.2 Software Configuration .... 10

1. **System Design**

4.1 Output Design ………………………………………………………………............. 14

* 1. Input Design …………………………………………………………………........... 14
  2. Database Design ………………………………………………………………...... 15
  3. Process Design ………………………………………………………...................... 15
  4. Structure Design ....................................................................................................... 16
  5. DFD Design ................................................................................................................ 16

1. **DFDs**

* 1. Level 0 .......................................................................................................................... 17
  2. Level 1 for Admin ........................................................................................................ 17
  3. Level 1 for customer .................................................................................................... 18
  4. Level 1 for Super Admin ............................................................................................. 18
  5. Loan Section ................................................................................................................ 19
  6. Stock Listing ................................................................................................................ 19
  7. Credit Card ................................................................................................................... 19
  8. Bank Account ................................................................................................................20

1. **System Implementation** 
   1. Introduction…........ 21
2. **Modules**

**7.1 Themes ...................................................................................................................... 22**

**7.2 Account ..................................................................................................................... 22**

**7.3 Loan products ........................................................................................................... 22**

**7.4 Blogs .......................................................................................................................... 23**

**7.5 Borrow Admin .......................................................................................................... 23**

**7.6 Super Admin ............................................................................................................. 23**

1. **Source Code ............................................................................................................. 24 - 41**
2. **System Testing**

**9.1 Purpose of testing .................................................................................................... 42**

**9.2 Types of testing ........................................................................................................ 43**

**9.3 Unit testing ............................................................................................................... 44**

**9.4 Integration testing .................................................................................................... 44**

**9.5 Program testing ........................................................................................................ 44**

**9.6 Sequential testing ..................................................................................................... 44**

**9.7 System testing ........................................................................................................... 45**

**9.9 Acceptance testing ................................................................................................... 45**

**9.10 Input testing ............................................................................................................. 45**

**9.11 Output testing .......................................................................................................... 46**

1. **System Maintenance**

**10.1 Corrective maintenance ........................................................................................ 47**

**10.2 Adaptive maintenance .......................................................................................... 47**

**10.3 Perfective maintenance ........................................................................................ 48**

**10.4 Preventive maintenance ....................................................................................... 48**

1. **Database Tables**

**11.1 User profile table ................................................................................................... 48**

**11.2 Loan table .............................................................................................................. 49**

**11.3 Login table ............................................................................................................. 50**

**11.4 Blog table ............................................................................................................... 50**

**11.5 Borrow admin ....................................................................................................... 51**

**11.6 Themes ................................................................................................................... 51**

1. **Sample Screen Shot ................................................................................................. 52 - 54**

**13** **Reference ........................................................................................................................ 55**

# **BORROW LOAN COMPANY**

**INTRODUCTION**

Borrow loan company is a Responsive and user-friendly web application for financial services aimed at simplifying the loan processing for both customers and financial institutions. The application offers a user-friendly interface that allows customers to easily apply for various types of loans, credit cards, stocks listing and account management. Key features include a comprehensive loan calculator, Blogs, Multi-Device Access, easy Loan Applications and secure document uploads. The platform also integrates with financial processing at back-end systems to e-mail based loan approval and disbursement processes. Through this innovative solution, it seeks to enhance the efficiency and accessibility of loan services, ultimately improving customer satisfaction and operational productivity for financial institutions.

**Goals**

* User-Friendly Interface: Develop an intuitive and easy-to-navigate interface that allows users to manage their loans effortlessly, whether they are borrowers, lenders, or administrators.
* Secure Data Management: Ensure the application implements robust security protocols to protect sensitive financial data, including encryption.
* Flexible Loan Structuring: Allow users to create and manage various loan types (e.g., fixed-rate, variable-rate, and interest-only loans) with customizable terms and conditions.
* Scalability and Performance: Design the application to handle a growing number of users and loans, with a scalable architecture that maintains performance and reliability as the user base expands.
* Mobile Accessibility: Develop a responsive design or mobile app that allows users to manage their loans from any device, ensuring accessibility on smartphones and tablets

## **SUMMARY**

Analysis is the process of breaking the problem into the successively manageable parts for individual study, system analysis is the study of various operations that has to be done to solve the problem .one aspect of the system analysis is defining the boundaries of the system and determining whether or not be proposed system should consider other related systems. One of the main meanings of the feasibility is possibility checking of the different criteria for success is included in feasibility study section .these criteria’s are cost ,time ,efficiency etc.…all these factors play an important role in achieving objective of the system .that means the system should be such it gives optimum performance at minimum cost ,time requirements .these system contributes to the overall objectives of the organization .the system be implemented using current technology and within given cost and schedule constrains .the system is integrated with systems which are already in place. It is a general term that refers to a structural process for identifying and solving problems .in a computer-based transformation system silk is the structured approach. Analysis implies the process of breaching something down in to its parts so that the whole may be understood. The definition of system analysis, but also that of synthesis, which is the process of putting parts together to form a new whole

**Proposed system**

If you have launched a loan processing system, then taking your loan processing in online can be the best business decision you will ever make. The Borrow Loan Company’s proposed system is an innovative and comprehensive solution designed to meet the needs of modern financial services. The proposed system is designed to simplify loan processing, ensure compliance with regulatory standards, and enhance user satisfaction while improving operational efficiency. By leveraging advanced technologies and integrating with existing financial systems, the platform aims to revolutionize loan services, increase customer satisfaction, and ensure operational excellence.

**Existing System**

The existing system is the manual system. Need to be converted into automated system. In the existing system contains lack of security risk to handling customer sensitive information's. Borrow Loan Company offers responsive and user-friendly web application designed to simplify financial services for both customers and financial institutions. This system focuses on improving the loan application process, ensuring secure data handling, and complying with regulatory requirements to provide efficient services to users.

## **FEASIBILITY STUDY**

Feasibility is conducted to identify the best system that meets all requirements. It is both necessary and important to evaluate the feasibility of a project at the earliest possible time. feasibility study includes an identification description, an evaluation of proposed system and selection of the best system for the job. During the system is to be carried out. this is to ensure that the proposed system is not A burden to the shop. The feasibility study should be relatively cheap and quick. the results should inform the decision of whether to go ahead with a more detailed analysis, some understanding of the major requirements for the system is essential. Four key considerations involved in the feasibility analysis are

* + 1. Technical feasibility
    2. Economical feasibility
    3. Behavioural feasibility
    4. Operational feasibility

#### **TECHNICAL FEASIBILITY**

A study of function, performance and constraints may improve the ability to create an acceptable system, technical feasibility is frequently the most difficult area to achieve at the stage of product Engineering process. technical feasibility is deals with the hardware as well as software requirements. the scope was whether the work for the project is done with the current equipment’s and the existing system technology has to be examined in the feasibility study. The result was found to be true. This feasibility is carried out to check the technical requirements of the system. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system. this is related to the technicality of the project. This evaluation determines whether the technology needed for proposed system is available or not .it deals with hardware as well as software requirements. that is, type of hardware, software and the methods required for running the system are analyzed .so it can be used in any windows so computer. This system requires very low system resources and it will work in almost all configurations. In the existing system all functions are doing manually. So, if they get this designed software, the problems can be avoided and thus the system will run smoothly.

#### **ECONOMICAL FEASIBILITY**

Economic feasibility is the most frequently used method for evaluating the effectiveness of the candidate system .it is very essential because the main goal of the proposed system is to have economically better result along with increased efficiency. A cost evaluation is weighed against the ultimate income or product. Economic justification is generally the bottomline consideration that includes cost benefit analysis, long term corporate income strategies, and cost of resources needed for development and potential market growth. When compared to the advantage obtained from implementing the system its cost is affordable. Proposed system was developed with available resources. Since cost input for the software is almost nil the output of the software is always a profit. Hence software is economically feasible.

#### **BEHAVIOURAL FEASIBILITY**

People are inherently resistant to change and computer is known for facilitating the changes an estimate should be made of how strongly the user; staff reacts towards the development of the computerized system. In the existing system more manpower is required and time factor is more. The more manpower for managing many files for dynamic data replication and more time for search through these files is needed. but in the proposed system, both manpower and time factors are reduced and also unnecessary burden is reduced. thus, the remaining people are made to engage in some other important work. also, there is no need to wait in case of downloading the data for the users therefore, the system is behaviorally feasible.

#### **OPERATIONAL FEASIBILITY**

The purpose of the operational feasibility is to determine whether the new system will be used if it is developed and implemented and whether there will be resistance from users that will undermine the possible application benefits .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 user 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 . The proposed system is an upgrade version of the current systems new fields have been implemented according to the user need, hence it ensures all the aspects.

## **TECHNOLOGY**

**Python**

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

**Hypertext Mark-up Language (HTML)**

It is the standard mark-up language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. HTML can embed programs written in a scripting language such as JavaScript, which affects the behaviour and content of web pages. Inclusion of CSS defines the look and layout of content

**MySQL Server**

MySQL is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius’s daughter, and "SQL", the abbreviation for Structured Query Language.

MySQL is free and open-source software under the terms of the GNU General Public License, and is also available under a variety of proprietary licenses. MySQL was owned and sponsored by the Swedish company MySQL AB, which was bought by Sun Microsystems (now Oracle Corporation MySQL is a component of the LAMP web application software stack (and others), which is an acronym for Linux, Apache, MySQL, Perl/PHP/Python. MySQL is used Features of SQL The major features of MySQL Server are:

* Easy installation
* Integration with internet
* Scalability and availabilty
* Support for client/server model
* Operating system capability
* Data replication support
* Full-text search
* Enterprise- level database
* Simplified database administration.

## **SYSTEM CONFIGURATION**

**HARDWARE CONFIGURATION:**

Selection of hardware configuration is very important task related to the software development. The processor should be powerful to handle all the operations. The hard disk should have the sufficient capacity to solve the database and the application.

### **HARDWARE SPECIFICATION:**

CPU - Pentium IV Processor And Above

Hard Disk - 40 GB

Display - 15”Color Monitor

Main Memory - 1GB

Keyboard - 104 Keys

Clock-Speed - 2.6 GHZ

Monitor - 15”CRT Monitor

### **SOFTWARE SPECIFICATIONS:**

Operating System - Windows 10

Web Server - Google Chrome, Internet Explorer, Microsoft Edge Environment - Python

Front End - HTML

Back End - MySQL SERVER

Browser - Internet Explorer 6 & Above

# **SYSTEM DESIGN**

The term design describes the final system and the way in which it is developed. The system design is a solution, how to approach to the new system. This important phase is composed of several steps. An emphasis is on translating the performance requirements of our proposed system into design specification. Design goes through logical and physical stage of development. In the design phase the physical design producing the working system by defining a particular specification that helps to knowing exactly what the new system must do. The logical design determines the information flow into and of the system and require database. Design is a multistep process that focuses on data structure, software, architecture, procedural details, and interface between modules. The design process translates the requirements into the representation of the software.

Computer software design changes continually because new methods, better analysis and broader understanding evolved. It provides the understanding and procedure details necessary for implementing the proposed system .an emphasis is on translating the performance requirement of our proposed system into design specification. Design goes through logical and physical stage. The system design is the last phase that indicate the final system and process of design phase. In the designed phase of maintenance management system, the database tables, input screens and output reports are designed. In table designing, redundancy is avoided. Design is the only way that we can accurately translate a system requirement into a software product. In our production management system, the all-input screens are designed as user friendly and understandable

## **INPUT DESIGN**

Input design is the link that ties the information system into the world of its users. The input design involves determining what the input is, how the data should be performed, how to validate data, how to minimize data entry and how to provide a multi user facility, inaccurate input data are the most common cause of errors in data processing. Errors entered by data entry operator can be controlled by input design. Input design is the process of converting user originated input to a computer-based format. Input data are collected and organized into groups of similar data. Once identified, appropriate input media are selected for processing

All the input data re validated in the order and if any data violates any conditions, the user is warned by a message. If the data satisfies all the conditions, then it is transferred to the appropriate table in the database. A form is designed to enter the details should be user friendlier so that authorized user with even less knowledge can enter the data. The form is designed using v b tools like command boxes, text boxes, labels, etc.

## **OUTPUT DESIGN**

Output design is very important concept in the computerized system, without reliable output the user may feel the entire system unnecessary and avoids using it. The proper output design is important in any system and facilitates effective decision making. The output design of this system includes various reports. output requirements are designed during system analysis. An application is successful only when it can provide efficient and effective reports.The goal of the output design is to capture the output and get the data into a format suitable for the computer. It is very helpful to produce the clear, accurate and speedy information for end users.A major form of the output is the harder copy from the pointer and screen reports. Printouts are designed around the output requirements of the user. Allowing the user to view the sample screen is important because the user is the ultimate judge of the quality of output. Output of this project is provided in the form of reports created using crystal report tool.

## **DATABASE DESIGN**

Database is a collection of interrelated data stores with minimum the overall objective in the development of the database technology has been to treat data as an organizational resource and has an integrated whole. Database management system allows data to be protected and organized separately from other resources. Database is an integrated collection of data. this is the difference between logical and physical data. The general objective is to make information access easy, quick, inexpensive and flexible for users. the database approach to system design places greater emphasis on the integration, integrity and independence of data.

## **PROCESS DESIGN**

Process design represents the structure of data and program components that are required to build a computer- based system. It considered the architectural style that the system will take, the structure and properties of the components constitute the system, and the interrelationships that occur among all architectural components of a system. Although a software engineer candesign both and architecture, the job is often allocated to specialist when large, complex system are to be built. A database or data warehouse designer creates data architecture for a system. The ‘system architect ‘select an appropriate architectural style for the requirements derived during system engineering and software requirement analysis. Architectural design begins with data design and proceeds to the derivation of one or more representations of the architectural structure of the system. An architecture model encompassing data architecture and program structure is created during architectural design. In addition, component properties and the process by which it is developed. It refers to technical specifications that will be applied in implementing the system. It includes the construction of program and program testing. The input to design phase is software requirement specification

## **STRUCTURE DESIGN**

Structured design deals with the data-flow in the system. It partitions a program into hierarchy of modules. The modules are organized in a top-down manner and the details will be at the bottom. The structured Design begins with a system specification that identifies inputs and outputs that described the functional of the Table.

## **DATAFLOW DIAGRAM DESIGN**

Data flow diagram (DFD) is a graphical representation of the “flow” of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. A DED shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored.

DFD is a designing tool used in the top-down approach to system Design. This context level DFD is next “exploded “, to produce a Level 1 DFD that shows some of the detail of the system being modelled. The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present in order for the system to do its job and shows the flow of data between the various parts of the system.

* + - Data Store- collection of data that is permanently stored.
    - External Entity- A person, organization or system that is external to the system but interact with it.
    - Data Flow- Single piece of data or logical collection of information like a bill.

The following are some DFD symbols used in the project.

Unable to load the shape

Rectangle: - It defines a source or destination of system data.

Unable to load the graphic

Circle: - It represents a process that transforms incoming data flow into outgoing data flow.

Unable to load the shape

Arrow: - It defines data flow. It is a pipeline through which information flows.

Unable to load the shape

Open rectangle: - It is used to store data or a temporary repository of data

Unable to load the graphic

Circle: - It represents a process that transforms incoming data flow into outgoing data flow.

Unable to load the shape

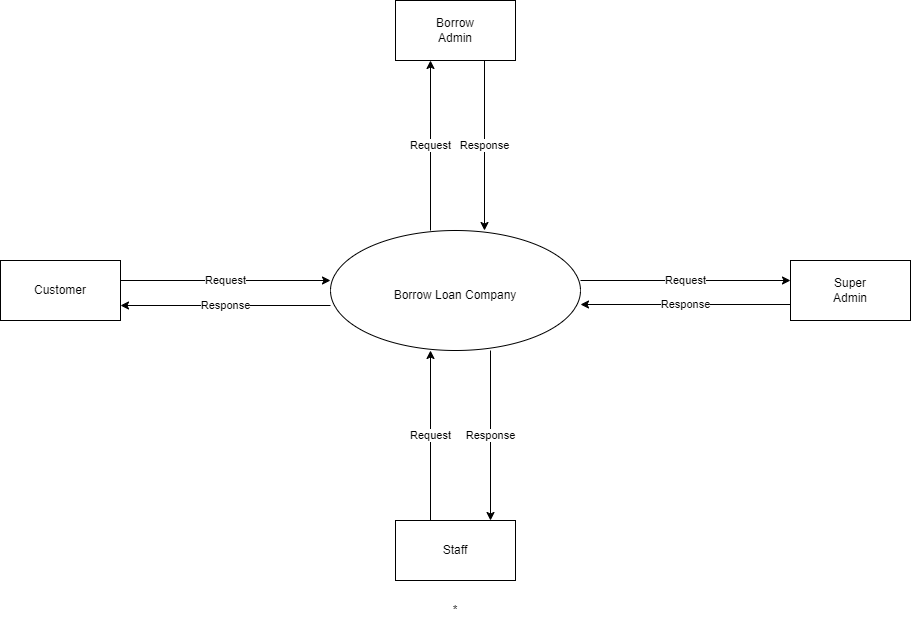
Arrow: - It defines data flow. It is a pipeline through which information flows.

Unable to load the shape

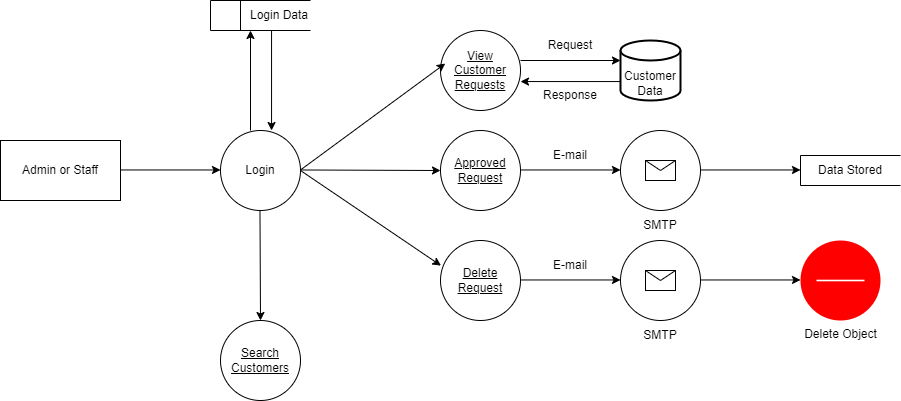
Open rectangle: - It is used to store data or a temporary repository of data

**DATA FLOW DIAGRAMS**

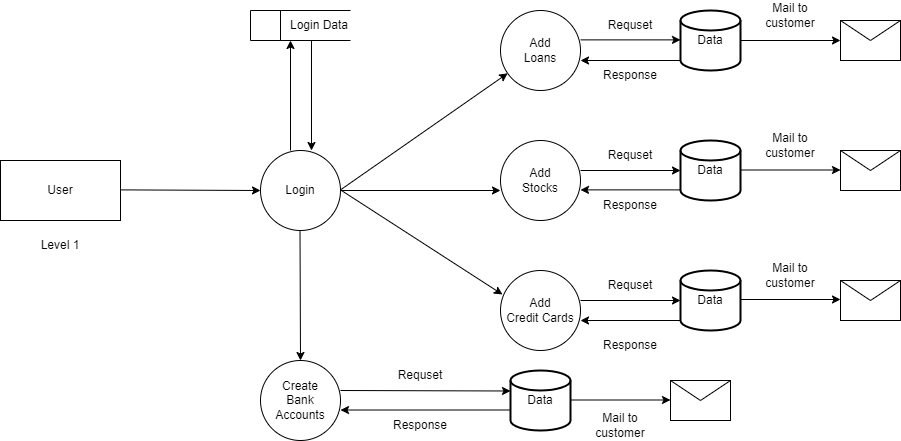
**Level 0**



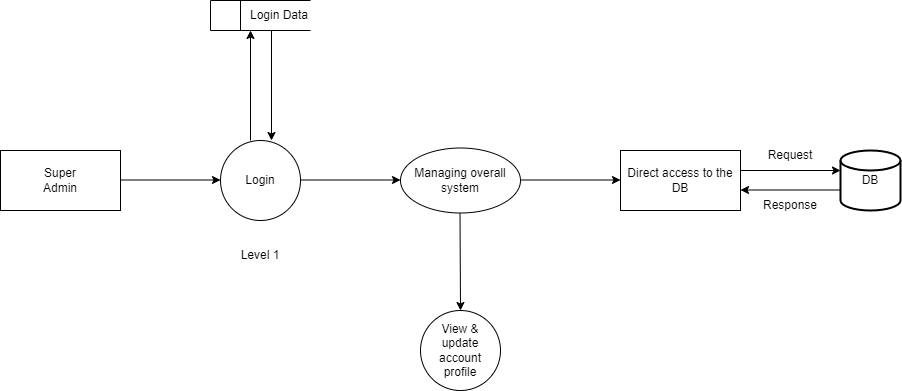
**Level 1 for admin**



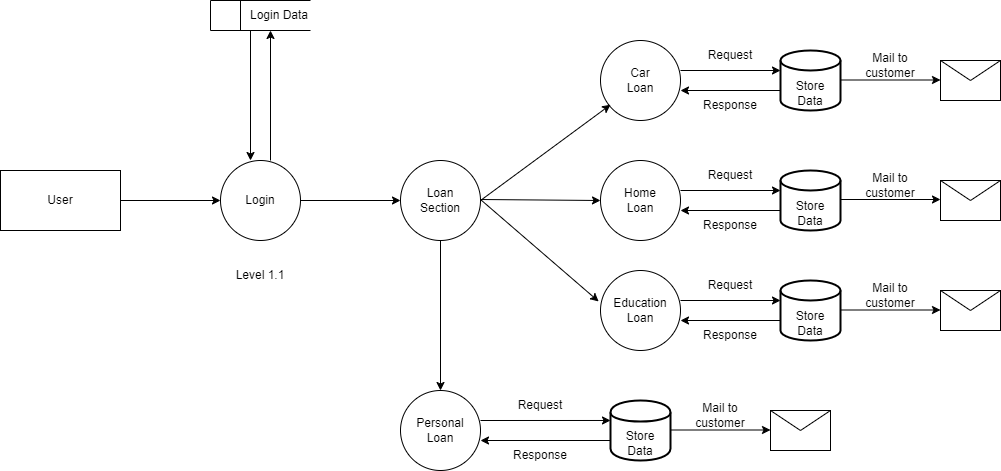
**Level 1 for customer**



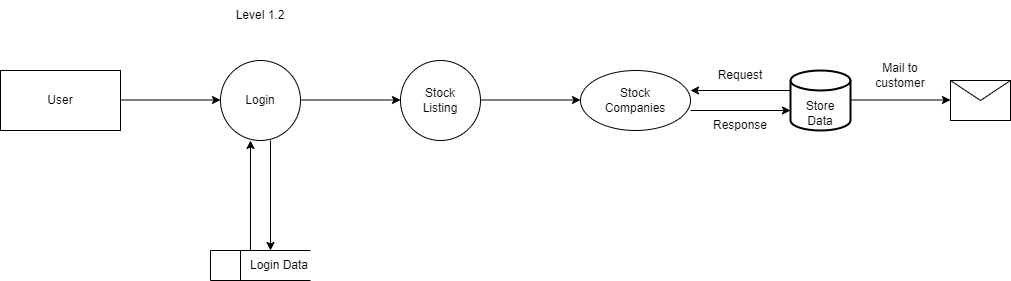
**Level 1 for super admin**



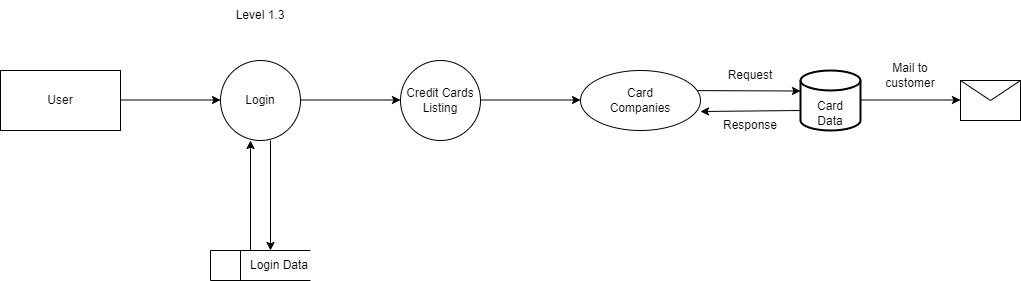
**Loan section DFD**



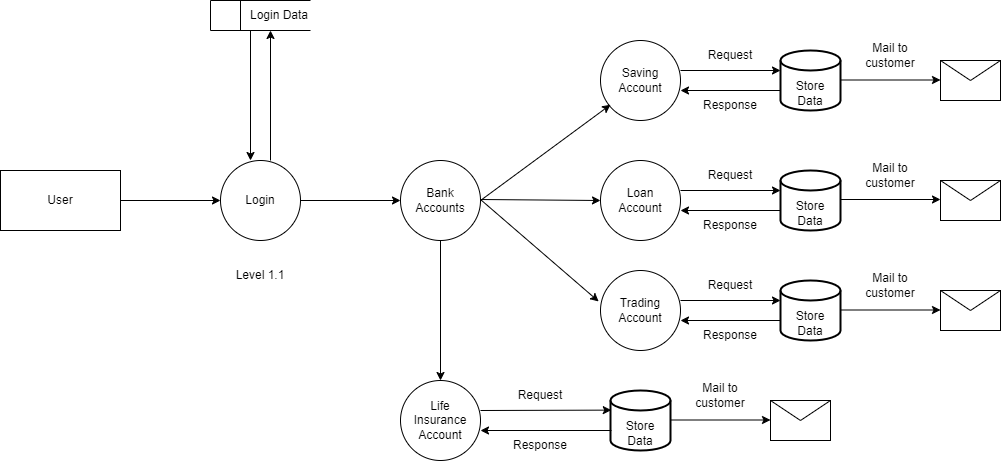
**Stock Listing DFD**



**Credit Card DFD**



**Bank Account DFD**



# **SYSTEM IMPLEMENTATION**

A crucial phase in the system life cycle is the successful implementation of the new system design. Implementation involves creating computer compatible files, training the operating staff, installing hardware, terminals. In the system implementation, user training is crucial for minimizing resistance to change and giving the new system a chance to prove its worth. The objectives of the system implementation are to put the system into operation while holding costs, risks and personal irritation to minimum. Once the physical system has been designed in details, the next stage is to run the design into a working system and then to monitor the operation of the system to ensure that is continue to work efficiently and the operation of the system to ensure that is continues to work efficiently and effectively. The implementation stage of is often very complex and time consuming because many more people are involved than in the earlier stages. The system implementation took place through various stages as follows,

* + - Implantation planning.
    - Education and training.
    - System testing.
    - System implementation.
    - Change over.

The implementation plan includes a description of all the activities that must occurs to implement the new system and to put it into operation. To achieve the objectives and benefits from computerbased system, it is essential for the people who will be confident of their role in the new jobs. After software is developed to meet user’s requirements, users test it for acceptance. The changes over phase are used to provide adaptability for the new system.

Module Specifications

* Themes
* Accounts
* Loan Products
* Blog Model
* Borrow Admin
* Super Admin

**Themes**

This module refers to the, design and templates of the web app, it ensuring a consistent and engaging user experience. This module supports a wide range of design elements, including color schemes, functions, layout configurations, and responsive design patterns, making it adaptable to various devices and screen sizes. It provides user-friendly interface and advanced customization options and also supports theme compatibility checks and real-time previews, ensuring smooth integration and a seamless aesthetic across the platform.

**Accounts**

This module refers to the, comprehensive functionalities for creating, maintaining, and closing different types of bank accounts, including savings, loan account, and fixed deposits. It enables operations such as deposits, withdrawals, balance inquiries, fund transfers, and account statements. The module incorporates security measures, such as encryption and authentication, to ensure the safety of financial transactions and customer data. The Bank Account Module is designed to be scalable, supporting integration with other banking services like loans and credit cards.

**Loan Products**

The Loan Products Module is a comprehensive solution designed to streamline and enhance various banking operations. It provides an integrated platform for loan processing, credit card processing and stock investing. By leveraging advanced technologies and user-friendly interfaces, the module aims to improve operational efficiency, reduce errors, and deliver a seamless experience for both bank staff and customers. Key features include real-time loan processing, automated compliance checks tools and secure data management. This module is tailored to meet the needs of modern banking institutions, ensuring scalability and adaptability in a rapidly evolving financial landscape.

**Blogs Model**

The Blogs Model Module is a robust and flexible component designed to facilitate the creation, management, and distribution of digital content across blogging platforms. This module offers a structured framework for handling various aspects of blogging, including post creation, content categorization, user interactions, and SEO optimization. It supports multimedia integration, customizable themes, and advanced publishing tools to enhance user engagement and content visibility. Additionally, the module incorporates analytics features to track performance metrics and audience behavior, empowering content creators with data-driven insights.

**Borrow Admin**

* Borrow admin can managing and monitoring the admin panel.
* The main activities of borrow admin were approving and rejecting loans, credit cards, and stock listings.
* Borrow Admin can make approved customers list.
* Borrow Admin can add, delete and update financial blogs.
* Borrow Admin can view customers financial Information's.

**Super Admin**

* Super Admin has the overall control of the system.
* Borrow Super admin can add and delete the Borrow Admins.
* Super Admin has separate admin panel, authentication and dashboard.
* Super admin can create new groups for Borrow Admin and Customers.
* Super Admin can selecting the borrow admin from the super admin panel.
* Super Admin can view customers account profile.

**Souce Code**

from django.shortcuts import redirect, render

from django.contrib.auth.models import User

from django.contrib.auth import login,logout,authenticate

from django.contrib import messages

from . forms import UserProfileForm

from . models import \*

from django.contrib.auth.decorators import login\_required

from . decorators import unauthaticated\_user

from django.core.mail import send\_mail

from django.conf import settings

# from django.contrib.auth.models import Group

# user account phase

@unauthaticated\_user

def sign\_up(request):

if request.method == 'POST':

try:

username = request.POST.get('username')

email = request.POST.get('email')

password = request.POST.get('password')

user\_data = User.objects.create\_user(username=username, email=email, password=password)

user\_data.save()

subject = 'Borrow Loan Company'

message = f'Dear {username},\nCongratulations on creating your new account with Borrow Loan Company. Thank you for choosing us for your finacial management.\nHere are your account details:\n\tUsername: {username}\n\tE-mail:{email}\nPlease keep this email for your records and do not forward or share any other person.\nTo get started, please visit our website at <https://borrowloancompany.pythonanywhere.com/> and log in with your new account details & Apply a new loan.\nWe look forward to serving you!'

recipient = email

send\_mail(subject,

message, settings.EMAIL\_HOST\_USER, [recipient], fail\_silently=True)

success\_msg = f'{username}, Account Registred Successfully'

messages.success(request, success\_msg)

return render(request, 'sign-in.html')

except Exception as e:

error\_msg = 'Username already exists try another username'

messages.error(request, error\_msg)

return render(request, 'sign-up.html')

return render(request, 'sign-up.html')

@unauthaticated\_user

def sign\_in(request):

if request.method == 'POST':

username = request.POST.get('username')

password = request.POST.get('password')

user\_auth = authenticate(request, username=username, password=password)

if user\_auth is not None:

login(request, user\_auth)

success\_msg = f'{username}, You are Logged In'

messages.success(request, success\_msg)

return render(request, 'home.html')

else:

error\_msg = 'Invalid username or password'

messages.error(request, error\_msg)

return render(request, 'sign-in.html')

return render(request, 'sign-in.html')

@login\_required(login\_url='sign\_in')

def sign\_out(request):

logout(request)

success\_msg = f'You are Logout Successfully'

messages.success(request, success\_msg)

return render(request, 'sign-in.html')

@login\_required(login\_url='sign\_in')

def account\_profile(request):

customer = request.user.userprofile

form = UserProfileForm(instance=customer)

if request.method == 'POST':

form = UserProfileForm(request.POST, request.FILES, instance=customer)

if form.is\_valid():

form.save()

username = request.user.username

success\_msg = f'{username}, Your profile updated successfully'

messages.success(request, success\_msg)

return redirect('account\_profile')

else:

form = UserProfileForm(instance=request.user.userprofile)

context = {'form': form}

return render(request, 'account-profile.html', context)

from django.shortcuts import render

from django.contrib import messages

from . models import \*

from django.contrib.auth.decorators import login\_required

from django.core.mail import send\_mail

from django.conf import settings

# Create your views here.

# loan proccessing phase

# for car\_loan

@login\_required(login\_url='sign\_in')

def car\_loan(request):

if request.method == 'POST':

try:

# getting personal details

loan\_type = request.POST.get('loan\_type')

name = request.POST.get('name')

email = request.POST.get('email')

phone = request.POST.get('phone')

address = request.POST.get('address')

age = request.POST.get('age')

dob = request.POST.get('dob')

country = request.POST.get('country')

photo = request.POST.get('photo')

id\_proof = request.POST.get('id\_proof')

# getting income details

employee\_id = request.POST.get('emp\_id')

employee\_name = request.POST.get('emp\_name')

job\_title = request.POST.get('job')

gross\_income = request.POST.get('gross\_income')

tax\_return = request.POST.get('tax\_return')

# getting loan details.

loan\_amount = request.POST.get('loan\_amount')

loan\_duration = request.POST.get('loan\_duration')

finacial\_statement = request.POST.get('finacial\_statement')

carloan\_data = CarLoan.objects.create(name=name, email=email, phone=phone, address=address, age=age, dob=dob,

country=country, photo=photo, id\_proof=id\_proof, emp\_id=employee\_id,

emp\_name=employee\_name, job\_title=job\_title, gross\_income=gross\_income,

tax\_return=tax\_return, loan\_amount=loan\_amount, loan\_duration=loan\_duration,

finacial\_statement=finacial\_statement, loan\_type=loan\_type)

carloan\_data.save()

subject = 'Borrow Loan Company Loan Section'

message = f'Dear {name},\nYou are successfully applied Car Loan with Borrow Loan Company.Our loan advisor will verify your documents and get in touch with you.\nHere are your car loan details:\n\tName: {name}\n\tEmail:{email}\n\tPhone Number: {phone}\n\tAddress: {address}\n\tLoan Amount: {loan\_amount}\n\tLoan Duration :{loan\_duration}\nPlease keep this email for your records and do not forward or share any other person.\nTo get started, please visit our website at <https://borrowloancompany.pythonanywhere.com/> and log in with your new account details.\nFor more details login with Borrow Loan Company.'

recipient = email

send\_mail(subject,

message, settings.EMAIL\_HOST\_USER, [recipient], fail\_silently=True)

success\_msg = 'Registered Successfully'

messages.success(request, success\_msg)

return render(request, 'register.html')

except Exception as e:

error\_msg = 'Error occured !'

messages.error(request,error\_msg)

return render(request, 'error.html')

return render(request, 'car-loan.html')

# for education\_loan

@login\_required(login\_url='sign\_in')

def education\_loan(request):

if request.method == 'POST':

try:

# getting personal details

loan\_type = request.POST.get('loan\_type')

name = request.POST.get('name')

email = request.POST.get('email')

phone = request.POST.get('phone')

address = request.POST.get('address')

age = request.POST.get('age')

dob = request.POST.get('dob')

country = request.POST.get('country')

photo = request.POST.get('photo')

id\_proof = request.POST.get('id\_proof')

# getting income details

cosig\_name = request.POST.get('co\_name')

cosig\_email = request.POST.get('co\_email')

cosig\_age = request.POST.get('co\_age')

cosig\_phone = request.POST.get('co\_phone')

cosig\_income = request.POST.get('co\_income')

cosig\_credit\_his = request.POST.get('co\_cre\_his')

# course details

course\_name = request.POST.get('course\_name')

course\_fee = request.POST.get('course\_fee')

course\_year = request.POST.get('course\_year')

# getting loan details.

loan\_amount = request.POST.get('loan\_amount')

loan\_duration = request.POST.get('loan\_duration')

educationloan\_data = EducationLoan.objects.create(name=name, email=email, phone=phone, address=address, age=age, dob=dob,

country=country, photo=photo, id\_proof=id\_proof, co\_signer\_name=cosig\_name,

co\_signer\_email=cosig\_email, co\_signer\_age=cosig\_age, co\_signer\_phone=cosig\_phone,

co\_signer\_income=cosig\_income, co\_signer\_credit\_history=cosig\_credit\_his, course\_name=course\_name,

course\_fee=course\_fee,course\_year=course\_year,loan\_amount=loan\_amount, loan\_duration=loan\_duration,

loan\_type=loan\_type)

educationloan\_data.save()

subject = 'Borrow Loan Company Loan Section'

message = f'Dear {name},\nYou are successfully applied Education Loan with Borrow Loan Company.Our loan advisor will verify your documents and get in touch with you.\nHere are your education loan details:\n\tName: {name}\n\tEmail:{email}\n\tPhone Number: {phone}\n\tAddress: {address}\n\tLoan Amount: {loan\_amount}\n\tLoan Duration :{loan\_duration}\nPlease keep this email for your records and do not forward or share any other person.\nTo get started, please visit our website at <https://borrowloancompany.pythonanywhere.com/> and log in with your new account details.\nFor more details login with Borrow Loan Company.'

recipient = email

send\_mail(subject,

message, settings.EMAIL\_HOST\_USER, [recipient], fail\_silently=True)

success\_msg = 'Registered Successfully'

messages.success(request, success\_msg)

return render(request, 'register.html')

except Exception as e:

error\_msg = 'Error occured !'

messages.error(request,error\_msg)

return render(request, 'error.html')

return render(request, 'education-loan.html')

# for home\_loan

@login\_required(login\_url='sign\_in')

def home\_loan(request):

if request.method == 'POST':

try:

# getting personal details

loan\_type = request.POST.get('loan\_type')

name = request.POST.get('name')

email = request.POST.get('email')

phone = request.POST.get('phone')

address = request.POST.get('address')

age = request.POST.get('age')

dob = request.POST.get('dob')

country = request.POST.get('country')

photo = request.POST.get('photo')

id\_proof = request.POST.get('id\_proof')

# getting income details

employee\_id = request.POST.get('emp\_id')

employee\_name = request.POST.get('emp\_name')

job\_title = request.POST.get('job')

gross\_income = request.POST.get('gross\_income')

tax\_return = request.POST.get('tax\_return')

# getting loan details.

loan\_amount = request.POST.get('loan\_amount')

loan\_duration = request.POST.get('loan\_duration')

finacial\_statement = request.POST.get('finacial\_statement')

homeloan\_data = HomeLoan.objects.create(name=name, email=email, phone=phone, address=address, age=age, dob=dob,

country=country, photo=photo, id\_proof=id\_proof, emp\_id=employee\_id,

emp\_name=employee\_name, job\_title=job\_title, gross\_income=gross\_income,

tax\_return=tax\_return, loan\_amount=loan\_amount, loan\_duration=loan\_duration,

finacial\_statement=finacial\_statement, loan\_type=loan\_type)

homeloan\_data.save()

subject = 'Borrow Loan Company Loan Section'

message = f'Dear {name},\nYou are successfully applied Home Loan with Borrow Loan Company.Our loan advisor will verify your documents and get in touch with you.\nHere are your home loan details:\n\tName: {name}\n\tEmail:{email}\n\tPhone Number: {phone}\n\tAddress: {address}\n\tLoan Amount: {loan\_amount}\n\tLoan Duration :{loan\_duration}\nPlease keep this email for your records and do not forward or share any other person.\nTo get started, please visit our website at <https://borrowloancompany.pythonanywhere.com/> and log in with your new account details.\nFor more details login with Borrow Loan Company.'

recipient = email

send\_mail(subject,

message, settings.EMAIL\_HOST\_USER, [recipient], fail\_silently=True)

success\_msg = 'Registered Successfully'

messages.success(request, success\_msg)

return render(request, 'register.html')

except Exception as e:

error\_msg = 'Error occured !'

messages.error(request,error\_msg)

return render(request, 'error.html')

return render(request, 'home-loan.html')

# for personal\_loan

@login\_required(login\_url='sign\_in')

def personal\_loan(request):

if request.method == 'POST':

try:

# getting personal details

loan\_type = request.POST.get('loan\_type')

name = request.POST.get('name')

email = request.POST.get('email')

phone = request.POST.get('phone')

address = request.POST.get('address')

age = request.POST.get('age')

dob = request.POST.get('dob')

country = request.POST.get('country')

photo = request.POST.get('photo')

id\_proof = request.POST.get('id\_proof')

# getting income details

employee\_id = request.POST.get('emp\_id')

employee\_name = request.POST.get('emp\_name')

job\_title = request.POST.get('job')

gross\_income = request.POST.get('gross\_income')

tax\_return = request.POST.get('tax\_return')

# getting loan details.

loan\_amount = request.POST.get('loan\_amount')

loan\_duration = request.POST.get('loan\_duration')

finacial\_statement = request.POST.get('finacial\_statement')

collateral = request.POST.get('collateral')

cosigner\_name = request.POST.get('co-name')

cosigner\_cred\_his = request.POST.get('co-his')

personalloan\_data = PersonalLoan.objects.create(name=name, email=email, phone=phone, address=address, age=age, dob=dob,

country=country, photo=photo, id\_proof=id\_proof, emp\_id=employee\_id,

emp\_name=employee\_name, job\_title=job\_title, gross\_income=gross\_income,

tax\_return=tax\_return, loan\_amount=loan\_amount, loan\_duration=loan\_duration,

finacial\_statement=finacial\_statement,collateral=collateral,co\_signer\_name=cosigner\_name,

co\_signer\_credit\_history=cosigner\_cred\_his, loan\_type=loan\_type)

personalloan\_data.save()

subject = 'Borrow Loan Company Loan Section'

message = f'Dear {name},\nYou are successfully applied Personal Loan with Borrow Loan Company.Our loan advisor will verify your documents and get in touch with you.\nHere are your personal loan details:\n\tName: {name}\n\tEmail:{email}\n\tPhone Number: {phone}\n\tAddress: {address}\n\tLoan Amount: {loan\_amount}\n\tLoan Duration :{loan\_duration}\nPlease keep this email for your records and do not forward or share any other person.\nTo get started, please visit our website at <https://borrowloancompany.pythonanywhere.com/> and log in with your new account details.\nFor more details login with Borrow Loan Company.'

recipient = email

send\_mail(subject,

message, settings.EMAIL\_HOST\_USER, [recipient], fail\_silently=True)

success\_msg = 'Registered Successfully'

messages.success(request, success\_msg)

return render(request, 'register.html')

except Exception as e:

error\_msg = 'Error occured !'

messages.error(request,error\_msg)

return render(request, 'error.html')

return render(request, 'personal-loan.html')

def register\_page(request):

return render(request,'register.html')

def error(request):

return render(request,'error.html')

# loan information phase

def car\_loan\_info(request):

return render(request,'car-loan-info.html')

def education\_loan\_info(request):

return render(request,'education-loan-info.html')

def home\_loan\_info(request):

return render(request,'home-loan-info.html')

def personal\_loan\_info(request):

return render(request,'personal-loan-info.html')

def personal\_loan\_lender(request):

return render(request,'personal-loan-lenders.html')

def student\_loan\_lender(request):

return render(request,'student-loan-lenders.html')

# credit card phase

def terms\_and\_conditions(request):

return render(request,'terms-and-conditions.html')

def credit\_card\_listing(request):

credit\_card = {

'credit\_card\_info':Credit\_Card.objects.all()

}

return render(request,'credit-card-listing.html',credit\_card)

def credit\_card\_details(request,pk):

credit\_card\_details = {

'credit\_card\_details':Credit\_Card.objects.get(pk=pk)

}

return render(request,'credit-card-details.html',credit\_card\_details)

@login\_required(login\_url='sign\_in')

def credit\_card\_form(request,pk):

card\_info = {

'card\_info':Credit\_Card.objects.get(pk=pk)

}

return render(request,'credit-card-form.html',card\_info)

@login\_required(login\_url='sign\_in')

def credit\_card\_data(request):

# fetching credit card data from customers

if request.method == 'POST':

try:

# getting personal details

customer\_photo = request.POST.get('customer\_photo')

full\_name = request.POST.get('fullname')

email = request.POST.get('email')

phone = request.POST.get('phone')

age = request.POST.get('age')

dob = request.POST.get('dob')

address = request.POST.get('address')

# getting finacial details

income = request.POST.get('income')

emp\_status = request.POST.get('emp\_status')

# occupation = request.POST.get('occupation')

emp\_name = request.POST.get('emp\_name')

emp\_address = request.POST.get('emp\_address')

bank\_account\_info = request.POST.get('bank\_account\_info')

# getting credit history details

credit\_histoty = request.POST.get('credit\_history')

id\_proof = request.POST.get('id\_proof')

card\_type = request.POST.get('card\_type')

credit\_card\_data = Credit\_Card\_Data.objects.create(full\_name=full\_name, email=email, phone=phone, dob=dob, customer\_age=age, address=address,

income=income, emp\_status=emp\_status, emp\_name=emp\_name,

emp\_address=emp\_address, bank\_account\_info=bank\_account\_info, credit\_history=credit\_histoty,

id\_proof=id\_proof, card\_type=card\_type, user\_photo=customer\_photo)

credit\_card\_data.save()

subject = 'Borrow Loan Company Credit Card Section'

message = f'Dear {full\_name},\nYou are successfully applied {card\_type} with Borrow Loan Company.Our credit card advisor will verify your documents and get in touch with you.\nHere are your card details:\n\tName: {full\_name}\n\tEmail:{email}\n\tPhone Number: {phone}\n\tAddress: {address}\n\tCard Type: {card\_type}\n\tCard Amount :50,000/-\nPlease keep this email for your records and do not forward or share any other person.\nTo get started, please visit our website at <https://borrowloancompany.pythonanywhere.com/> and log in with your new account details.\nFor more details login with Borrow Loan Company.'

recipient = email

send\_mail(subject,

message, settings.EMAIL\_HOST\_USER, [recipient], fail\_silently=True)

success\_msg = 'Registered Successfully'

messages.success(request, success\_msg)

return render(request,'successfully.html')

except Exception as e:

error\_msg = 'Error occured !'

messages.error(request,error\_msg)

return render(request,'error.html')

def successfully(request):

return render(request,'successfully.html')

# stock invest and firms phase

def stock\_listing(request):

stock\_vendors = {

'stock\_vendors':Stock\_Company\_Vendors.objects.all()

}

return render(request,'stock-listing.html',stock\_vendors)

def stock\_info(request,pk):

stock\_info = {

'stock\_firm':Stock\_Company\_Vendors.objects.get(pk=pk)

}

return render(request,'stock-info.html',stock\_info)

@login\_required(login\_url='sign\_in')

def stock\_invest\_form(request,pk):

stock\_form = {

'stock\_form':Stock\_Company\_Vendors.objects.get(pk=pk)

}

return render(request,'stock-invest-form.html',stock\_form)

@login\_required(login\_url='sign\_in')

def stock\_invest\_data(request):

if request.method == 'POST':

try:

# getting personal details

customer\_photo = request.POST.get('customer\_photo')

full\_name = request.POST.get('fullname')

email = request.POST.get('email')

phone = request.POST.get('phone')

dob = request.POST.get('dob')

address = request.POST.get('address')

income = request.POST.get('income')

# getting account info

account\_num = request.POST.get('acc\_num')

account\_username = request.POST.get('acc\_username')

# getting tax and SSN info

soc\_sec\_num = request.POST.get('soc\_sec\_num')

tax\_status = request.POST.get('tax\_status')

# getting stock data

invest\_amount = request.POST.get('invest\_amount')

fund\_source = request.POST.get('fund\_source')

stock\_company\_vendor = request.POST.get('type')

stock\_data = Stock\_Invest\_Data(full\_name=full\_name, email=email, phone=phone, dob=dob, address=address,

income=income, account\_number=account\_num, account\_username=account\_username,

social\_security\_number=soc\_sec\_num, tax\_status=tax\_status, invest\_amount=invest\_amount,

source\_of\_invest=fund\_source, stock\_company\_vendor=stock\_company\_vendor, user\_photo=customer\_photo)

stock\_data.save()

subject = 'Borrow Loan Company Stock Invest'

message = f'Dear {full\_name},\nYou are successfully invest {stock\_company\_vendor} stocks with Borrow Loan Company.Our market advisor will verify your documents and get in touch with you.\nHere are your stock details:\n\tName: {full\_name}\n\tEmail:{email}\n\tPhone Number: {phone}\n\tAddress: {address}\n\tCompany Revenue: 2cr/month\n\tPurchased Amount :{invest\_amount}\nPlease keep this email for your records and do not forward or share any other person.\nTo get started, please visit our website at <https://borrowloancompany.pythonanywhere.com/> and log in with your new account details.\nFor more details login with Borrow Loan Company.'

recipient = email

send\_mail(subject,

message, settings.EMAIL\_HOST\_USER, [recipient], fail\_silently=True)

success\_msg = 'Registered Successfully'

messages.success(request, success\_msg)

return render(request,'successfully.html')

except Exception as e:

error\_msg = 'Error occured !'

messages.success(request, error\_msg)

return render(request,'error.html')

def global\_stock\_rate(request):

return render(request, 'global-stock-rate.html')

Database code sample

from django.db import models

from django.contrib.auth.models import User

# Create your models here.

class UserProfile(models.Model):

user = models.OneToOneField(User, on\_delete=models.CASCADE)

profile\_picture = models.ImageField(default='media/-140246016.jpg', upload\_to='profile-photo/', blank=True, null=True)

user\_age = models.CharField(max\_length=20, default='Enter your Age', blank=True, null=True)

name = models.CharField(max\_length=50)

email = models.EmailField()

phone = models.CharField(default='Enter your Number', max\_length=20, blank=True, null=True)

address = models.CharField(default='Enter your Address', max\_length=100, blank=True, null=True)

country = models.CharField(default='Enter your Country', max\_length=50, blank=True, null=True)

dob = models.CharField(default='DD/MM/YYYY', max\_length=20, blank=True, null=True)

id\_proof = models.FileField(default='Upload your ID Proof', upload\_to='id-proof/', blank=True, null=True)

def \_\_str\_\_(self):

return f"{self.user.username} profile"

@allowed\_users(allowed\_roles=['admin'])

def loan\_appointment\_list(request):

loans = {

'carloan':CarLoan.objects.all(),

'educationloan':EducationLoan.objects.all(),

'homeloan':HomeLoan.objects.all(),

'personalloan':PersonalLoan.objects.all(),

}

return render(request, 'loan-appointment-list.html', loans)

@allowed\_users(allowed\_roles=['admin'])

def creditcard\_appointment\_list(request):

card\_data = {

'credit\_card':Credit\_Card\_Data.objects.all()

}

return render(request, 'creditcard-appointment-list.html', card\_data)

@allowed\_users(allowed\_roles=['admin'])

def stock\_invest\_list(request):

stock\_company\_data = {

'stock\_data':Stock\_Invest\_Data.objects.all()

}

return render(request, 'stock-invest-list.html', stock\_company\_data)

@allowed\_users(allowed\_roles=['admin'])

def approved\_list(request):

approved\_list ={

'approved\_loans':Approved\_Loans.objects.all(),

'approved\_cards':Approved\_Cards.objects.all(),

'approved\_stocks':Approved\_Stocks.objects.all(),

}

return render(request, 'approved-list.html', approved\_list)

@allowed\_users(allowed\_roles=['admin'])

def reviews(request):

feedback = {

'feedback':Feedback.objects.all()

}

return render(request, 'reviews.html', feedback)

@allowed\_users(allowed\_roles=['admin'])

def loan\_account\_list(request):

loanaccount = {

'data':LoanAccount.objects.all()

}

if request.method == 'POST':

search\_data = request.POST.get('q')

search\_result = LoanAccount.objects.filter(Q(fullname\_\_icontains=search\_data)|Q(email\_\_icontains=search\_data)|Q(account\_number\_\_icontains=search\_data)|Q(phone\_\_icontains=search\_data))

return render(request, 'loan-account-list.html', {'data':search\_result})

else:

return render(request, 'loan-account-list.html', loanaccount)

@allowed\_users(allowed\_roles=['admin'])

def trading\_account\_list(request):

tradingaccount = {

'data':TradingAccount.objects.all()

}

if request.method == 'POST':

search\_data = request.POST.get('q')

search\_result = TradingAccount.objects.filter(Q(fullname\_\_icontains=search\_data)|Q(email\_\_icontains=search\_data)|Q(account\_number\_\_icontains=search\_data)|Q(phone\_\_icontains=search\_data))

return render(request, 'trading-account-list.html', {'data':search\_result})

else:

return render(request, 'trading-account-list.html', tradingaccount)

@allowed\_users(allowed\_roles=['admin'])

def saving\_account\_list(request):

savingaccount = {

'data':SavingAccount.objects.all()

}

if request.method == 'POST':

search\_data = request.POST.get('q')

search\_result = SavingAccount.objects.filter(Q(fullname\_\_icontains=search\_data)|Q(email\_\_icontains=search\_data)|Q(account\_number\_\_icontains=search\_data)|Q(phone\_\_icontains=search\_data))

return render(request, 'saving-account-list.html', {'data':search\_result})

else:

return render(request, 'saving-account-list.html', savingaccount)

@allowed\_users(allowed\_roles=['admin'])

def lifeinsurance\_account\_list(request):

life\_insurance\_account = {

'data':Life\_Insurance\_Account.objects.all()

}

if request.method == 'POST':

search\_data = request.POST.get('q')

search\_result = Life\_Insurance\_Account.objects.filter(Q(fullname\_\_icontains=search\_data)|Q(email\_\_icontains=search\_data)|Q(account\_number\_\_icontains=search\_data)|Q(phone\_\_icontains=search\_data))

return render(request, 'lifeinsurance-account-list.html', {'data':search\_result})

else:

return render(request, 'lifeinsurance-account-list.html', life\_insurance\_account)

@allowed\_users(allowed\_roles=['admin'])

def carloan\_approved\_email(request, pk):

approved\_customer = {

'carloan\_customer':CarLoan.objects.get(pk=pk),

}

return render(request, 'carloan-approving-email.html', approved\_customer)

**System Testing**

Testing is an important phase in software development. After completion, the system may work without any problem. But there should be several unknown or hidden errors in the system still remaining. The error chances may be injected into the system at any stage of the development. Even if there are techniques to detect and eliminate the errors, some errors may retain in the system. So, after the completion of coding, the system is to be executed with the only purpose of detecting maximum number of errors. The tester executes the system, and inputs different types of values those may cause error or some exceptional situation in the system. The error locations detected through the testing are to be corrected in the system then. So, the important and the only aim of testing is to detect and cure even a less possible of an error that may face in the future executions of the system. Testing is a set of activity that can be planned in advance and conducted systematically. Testing begins at the module level and work towards the integration of entire computers-based system. Nothing is completed without testing, as it is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are corrected, the goal will be successfully achieved. Inadequate testing or non-testing may lead to errors that may not appear until months later.

## **PURPOSE OF TESTING**

Testing is the success of the system. System testing makes a logical assumption that if all part of the system is correct, the goal will be successfully achieved. The following points shows how testing is essential. Existence of program defects of inadequacies is inferred. Verifies whether the software behave as intended by its designer. Checks conformance with requirements specification or user need.

Access the operational reliability of the system. Test the performance of the system. The performance of the system. Reflects the frequencies of actual user inputs. Find the fault which caused the output anomaly. Detect flaws and deficiencies in requirements. Exercise the program using data like the real data processed by the program. Test the system capabilities. Judges whether or not the program is usable in practice. Testing objectives there are several rules that can serve as testing objectives. They are; Testing is a process of executing a program with the intent of finding error. A good test case is one that has high probability of finding an undiscovered error. A successful test is one that uncovers an undiscovered error.If testing is conducted successfully according to the objectives as stated above, it would uncover errors in the software. Also testing demonstrates that software functions appear to the working according to the specifications, that performance requirement appear to have been met. These are three ways to test a program for correctness for implementation efficiency for computational complexity Test for correctness are supported to verify that a program does exactly what it was designed to do. This is much difficult that it may at first appear especially for large programs. Tests for implementation efficiency attempt to find ways to make a correct program faster or use less storage. It is a code-refining process, which reexamines the implementation phase algorithm development. Tests for computational complexity amount to an experiment analysis of the complexity of an algorithm or an experiment comparison of two or more algorithms, which solve the same problem.

## **TYPES OF TESTING**

System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before live operation commences. Testing is vital to the success of the system. System testing makes a logical assumption that if all the parts of the system are correct. The goal will be successfully achieving. The candidate system is subject to a variety of tests. A series of tests are performed for the proposed system is ready for system acceptance testing. The various levels at which testing are conducted are,

* + 1. Unit testing
    2. Integration testing
    3. Sequential testing
    4. System testing
    5. Validation testing unit testing

## **UNIT TESTING**

In unit testing each program unit is tested individually.so any errors in a unit are debugged. Sample data is given for unit testing. The unit test results are recorded for future references. Unit testing focus verification efforts on the smallest unit of software design, the module. This is known as “module testing”. It comprises of the set test performed by an individual programmer prior to the integration of unit into the large system. The modules are tested separately, this testing is carried out programming stage itself.In this step each module is found to be working satisfactory as regard to the expected out from module. The unit testing was done for every module in the software for various inputs, such they each line of code is at least once executed. This testing was carried out during the unit to a large system.

## **INTEGRATION TESTING**

Integration testing is a systematic technique for constructing the program structure while at the same time conducting test to uncover errors associated with interfacing.

## **PROGRAM TESTING**

Program testing checks for two types of errors; syntax and logic. A syntax error is a program statement that violates one or more rules of the language in which it is written. A logic error deals with incorrect data fields. When a program is tested, the actual output is compared with the expected output. All the modules are combined and tested as a whole. Here correction is difficult because the vast expenses of all errors uncovered are correct for the next testing steps. We follow bottom-up integration. Bottom-up integration testing as its name implies begin construction and sling with atomic modules. Because components are integrated from the bottom up, accessing required for the components subordinate to a given level is always available and need for stubs is eliminated.

## **SEQUENTIAL TESTING**

Sequential or series testing is checking the logic of one or more programs in the candidate system, where the output of one program will affect the processing done by another program.

## **SYSTEM TESTING**

System testing executing a program to check logic changes made in it and with the intension of finding errors-making the program fails. Effective testing does not guaranty reliability is a design consideration. This testing actually consists of a series of different test whose primary purpose is to fully exercise the computer based system.it begins where integration testing is completed and finally software is completely assembled as package, interfacing errors are uncovered and corrected.

## **ACCEPTANCE TESTING**

Acceptance testing is running the system with live data by the actual user. An acceptance test has the objective of selling the user in the validity and reliability of the system. A comprehensive test report is prepared. The report indicates the system’s tolerance, performance range, error rate and accuracy. It verifies the system procedures operate to system specification and the integrity of important data is maintained, performance of an acceptance test is actually the users show. User motivation is very important for the successful performance of the system. After that a comprehensive test report is prepared. This report shows the systems tolerance, performance range, error rate and accuracy.

## **INPUT TESTING**

Here system is tested with all verifiable combination of input. User may type data in situations like entering password, numerical details etc. The system is tested with all the causes and it responded with appropriate error message.

## **OUTPUT TESTING**

Here the output is tested to view where the screen is what which is desired. It is also checked whether it is to the satisfaction of the user. Changes that need to be done can be done after the result is seen

**System Maintenace**

Software maintenance is the process of modifying a software system or component after its delivery in order to correct faults improve the performance and other attributes, or to adapt to the changed environment. maintenance covers a wide range of activities including correcting the error and design coding, updating the documentation and test data, and upgrading the user hardware and software. maintenance is always necessary to keep the software usable and useful. Hardware also requires periodic maintenance to keep the system into its standards. After installation is completed and user start is adjusted to the changes created by the candidate system. evaluation and maintenance begin. If new information is consistent with design specification the changes have to be made. Hardware also requires periodic maintenance to keep in tune with design specifications. User priorities changes in organizational requirements or environmental factors also called for system enhancements. Maintenance covers wide range of activities, including correcting, coding and design errors, updating documentation and test data, and upgrading user support. any activities classified as maintenance are actually enhancements. Maintenance means restoring something to do its original condition. Unlike hardware, software does not wear out; it is corrected.in contrast, enhancement means adding, modifying or redeveloping the code to support changes in the specifications.it is necessary to keep up with changing user needs the operational environment. Maintenance means repairing processing or performance failures or making changes because of previously uncorrected problems or false assumptions. adaptive maintenance means changing the program function. Perfective maintenance means enhancing the performance or modifying the program to respond to the user’s additional or changing needs. Of these types, more time and money are spent on perfective than on corrective and adaptive maintenance together. Maintenance activities begin where conversion leaves off. Maintenance is handled by the same planning and control used in a formal system project. a major problem with Software maintenance is its labor-intensive nature. Documentation is as much a part of maintenance as it is of system development .to put maintenance in its proper perspective requires considerable skill and experience and is an important and is an important and ongoing aspect of system development .an additional factor in the success of the maintenance programmer is the work environment.

**Maintenance Types**

**System maintenance can be classified into four types –**

* + - * Corrective Maintenance
      * Adaptive Maintenance
      * Perfective Maintenance
      * Preventive Maintenance

**CORRECTIVE MAINTENANCE**

Corrective Maintenance deals with the repair of faults or defects found in day- today system functions. A defect can result due to errors in software design, logic and coding. Design errors occur when changes made to the software are incorrect, incomplete, wrongly communicated, or the change request is misunderstood. Logical errors result from invalid tests and conclusions, incorrect implementation of design specifications, faulty logic flow, or incomplete implementation of design specifications, faulty logic flow, or incomplete test of data. All these errors, referred to as residual errors, prevent the software from confirming to its agreed specifications. Note that the need for corrective maintenance is usually initiated by big reports drawn by the users.

## **ADAPTIVE MAINTENANCE**

Adaptive Maintenance is the implementation of changes in a part of the system, which has been affected by a change that occurred in some other part of the system. Adaptive Maintenance consists of adapting software to changes in the environment such as the hardware or the operating system. The term environment in this context refers to the conditions and the influences which act (from outside) on the system. For example, business rules, work patterns and government policies have a significant impact on the software system.

## **PERFECTIVE MAINTENANCE**

Perfective Maintenance mainly deals with implementing new or changed user requirements. Perfective Maintenance involves making functional enhancements to the system in addition to the activities to increase the system’s performance even when the changes have not been suggested by faults. This includes enhancing both the function and efficiency of the code and changing the functionalities of the system as per the users’ changing needs.

## **PREVENTIVE MAINTENANCE**

Preventive Maintenance involves performing activities to prevent the occurrence of errors. It tends to reduce the software complexity thereby improving program understand ability and increasing software maintainability. It comprises documentation updating, code optimization and code restructuring. Documentation updating involves modifying the documents affected by the changes in order to correspond to the present state of the system. Code optimization involves modifying the programs for faster execution or efficient use of storage space. Code restructuring involves transforming the program structure for reducing the complexity in source code and making it easier to understand.

# **DATABASE TABLES:**

**UserProfile Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELDS** | **DATATYPE** | **CONSTRAINT** | **DESCRIPTION** |
| user\_id | Int() NOT NULL | Primary key | User profile id |
| user | Charfield(50) | Foreign key | Build relationship |
| User\_age | Charfield(50) | NULL | User age |
| name | Charfield(50) | NULL | username |
| email | Emailfield() | NULL | Email address |
| phone | Charfield(50) | NULL | User number |
| address | Charfield(50) | NULL | User address |
| country | Charfield(50) | NULL | User country |

**Loan Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELDS** | **DATATYPE** | **CONSTRAINT** | **DESCRIPTION** |
| Loan\_id | Int(100) | Primary key | Loan id |
| name | Charfield(100) | Not Null | Customer name |
| email | Emailfield() | Not Null | Customer email address |
| phone | Charfield(100) | Not Null | Customer phone number |
| age | Charfield(100) | Not Null | Customer age |
| dob | Charfield(100) | Not Null | Customer date of birth |
| country | Charfield(100) | Not Null | Customer country |
| Id\_proof | Filefield(100) | Not Null | Customer id proof |
| address | Charfield(100) | Not Null | Customer address |
| photo | Filefield(100) | Not Null | Customer photo |
| Emp\_id | Charfield(100) | Not Null | Customer employee id |
| Emp\_name | Charfield(100) | Not Null | Customer name |
| Job\_title | Charfield(100) | Not Null | Customer current job |
| Gross\_income | Filefield(100) | Not Null | Customer gross income |
| Tax\_return | Filefield(100) | Not Null | Customer tax return |
| Loan\_amount | Charfield(100) | Not Null | Customer loan amount |
| Loan\_duration | Charfield(100) | Not Null | Customer loan duration |
| Finacial\_statement | Filefield(100) | Not Null | Customer financial statement |
| Date\_time | Charfield(100) | Not Null | Date time field for processing loan request |
| Loan\_type | Charfield(100) | Not Null | Loan type for classification |

**Login**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **TYPE** | **CONSTRAINT** | **DESCRIPTION** |
| user\_id | Varchar(45) | Primary key | Id |
| Password | Varchar(45) | Null | password |

**Blogs**

|  |  |  |  |
| --- | --- | --- | --- |
| **FILED NAME** | **DATA TYPE** | **CONSTRAINT** | **DESCRIPTION** |
| Blog\_id | Int (100) | Primary key | Id |
| Blog\_pic | Filefield() | Not null | Blog image |
| Blog\_title | Charfield(100) | Not null | Blog heading |
| Blog\_desc | Textfield() | Not null | Blog description |
| Date\_time | Datetimefield() | Not null | Blog created date and time |

**Borrow Admin**

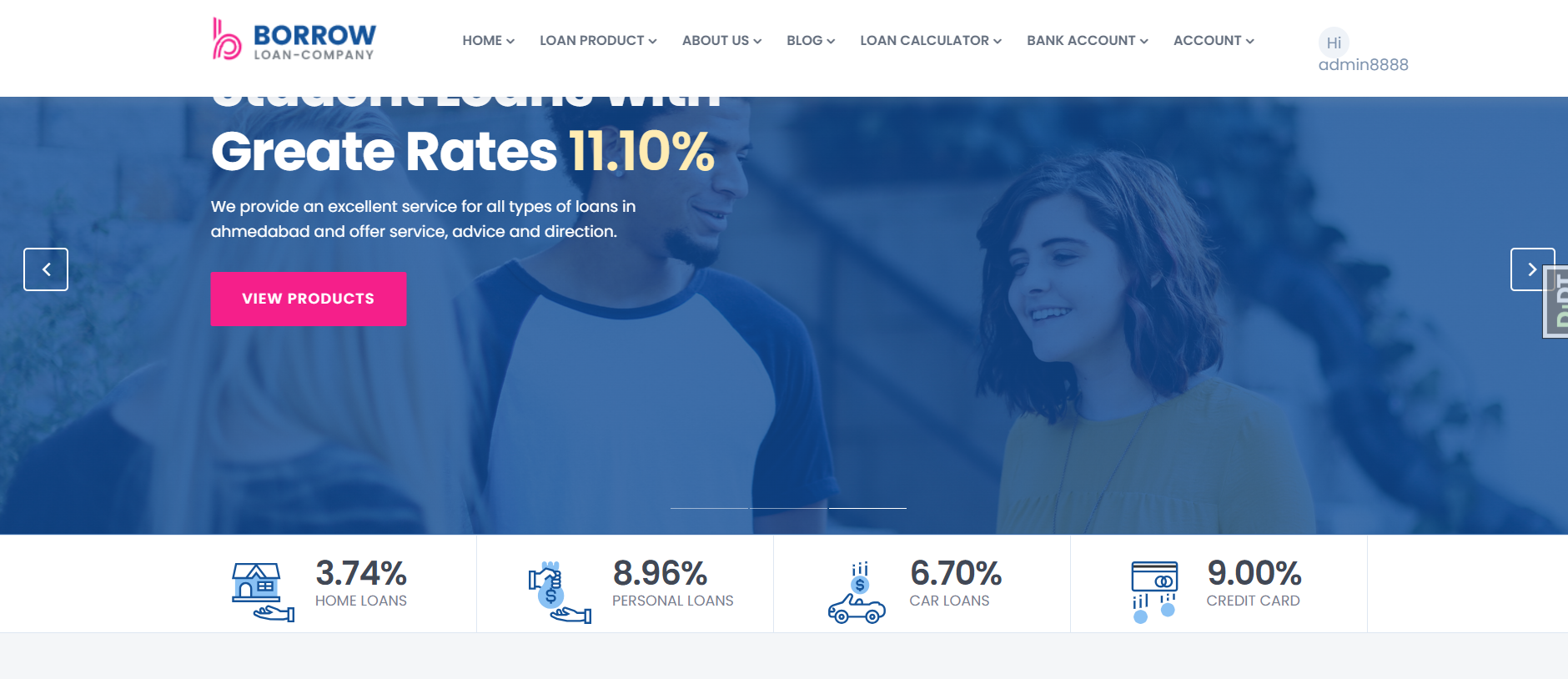
|  |  |  |  |
| --- | --- | --- | --- |
| FILED NAME | DATA TYPE | CONSTRAINT | DESCRIPTION |
| Cus\_id | Int (100) | Primary key | Id |
| Cus\_name | Filefield() | Not null | Blog image |
| Cus\_email | Charfield(100) | Not null | Blog heading |
| Cus\_phone | Textfield() | Not null | Blog description |
| Loan\_type | Charfield(100) | Not null | Loan type |
| Card\_type | Charfield(100) | Not null | Card type |
| Stock\_company | charfield(100) | Not null | Stock company |

**Themes**

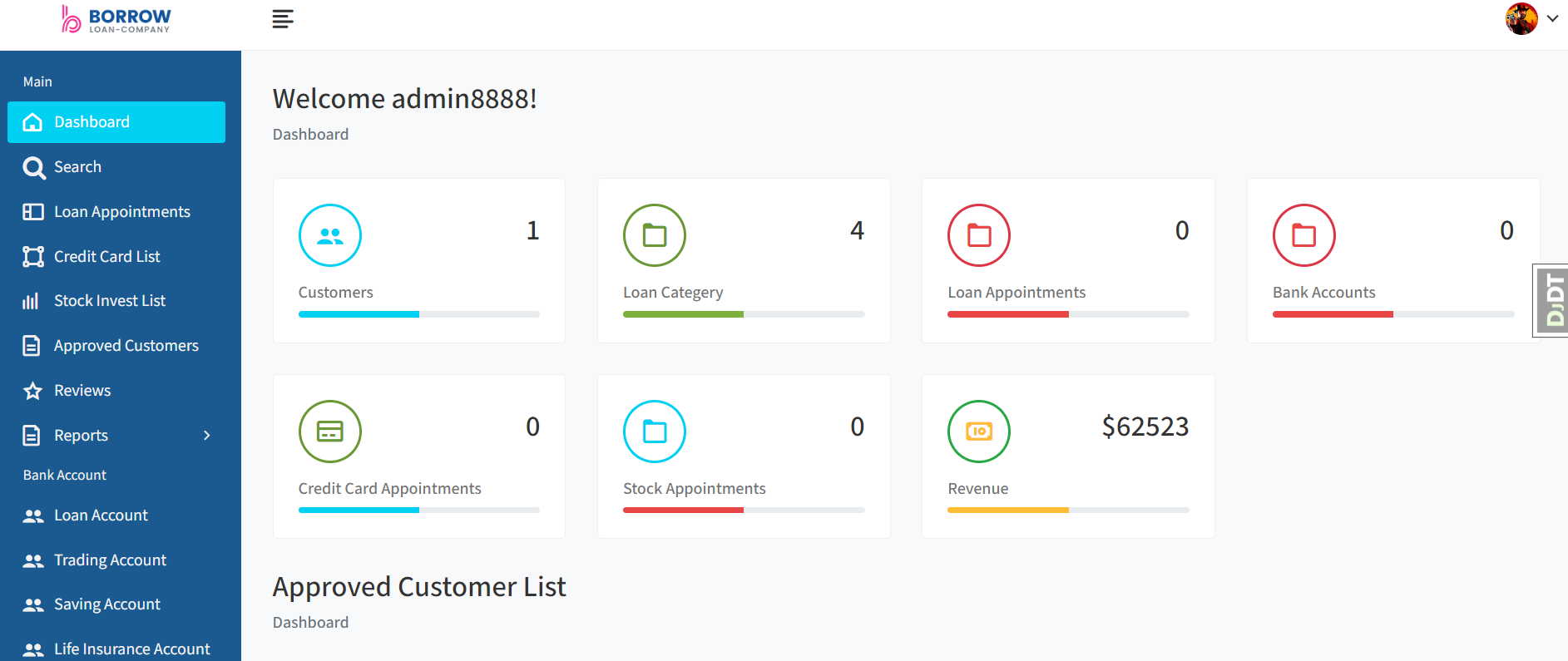
|  |  |  |  |
| --- | --- | --- | --- |
| FILED NAME | DATA TYPE | CONSTRAINT | DESCRIPTION |
| Feedback\_id | Int (100) | Primary key | Id |
| name | charfield(100) | Not null | Reviewer name |
| email | emailfield() | Not null | Reviewer email address |
| phone | charfield(10) | Not null | Reviewer phone number |
| Date\_time | Datetimefield() | Not null | created date and time |

**Sample Screen Shoots**

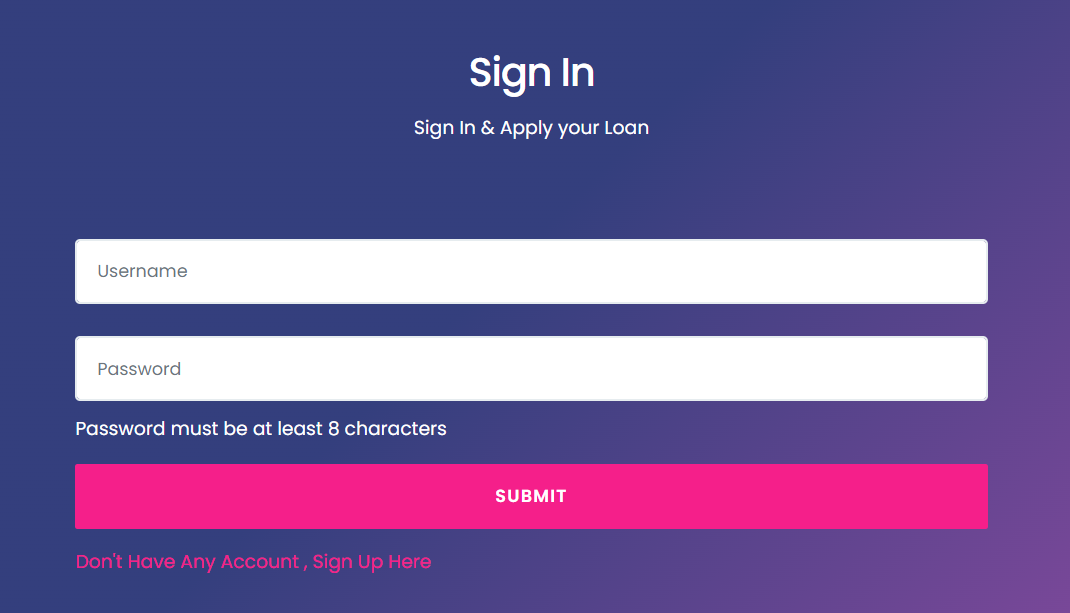
**Home Page**

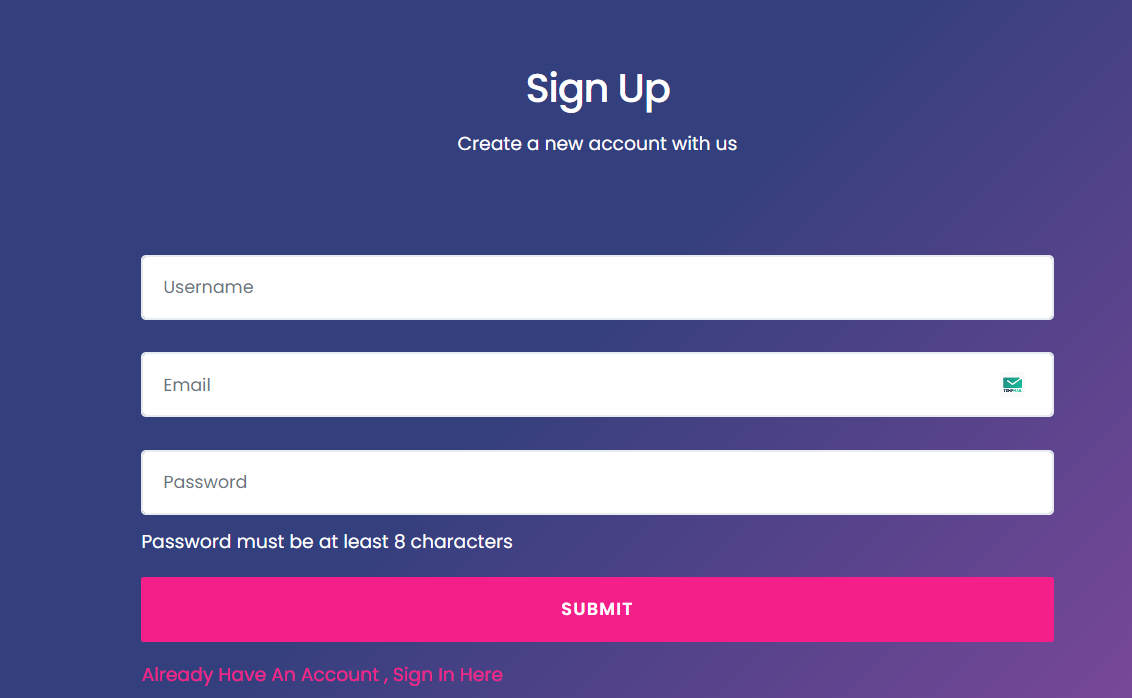


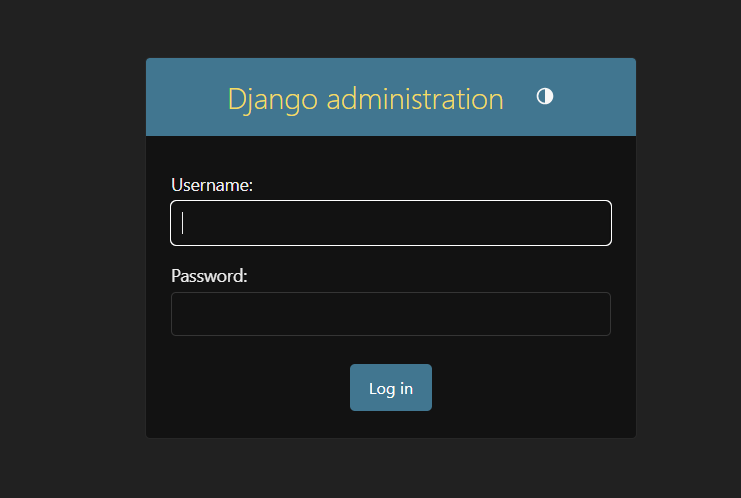
**Admin Dashboard**

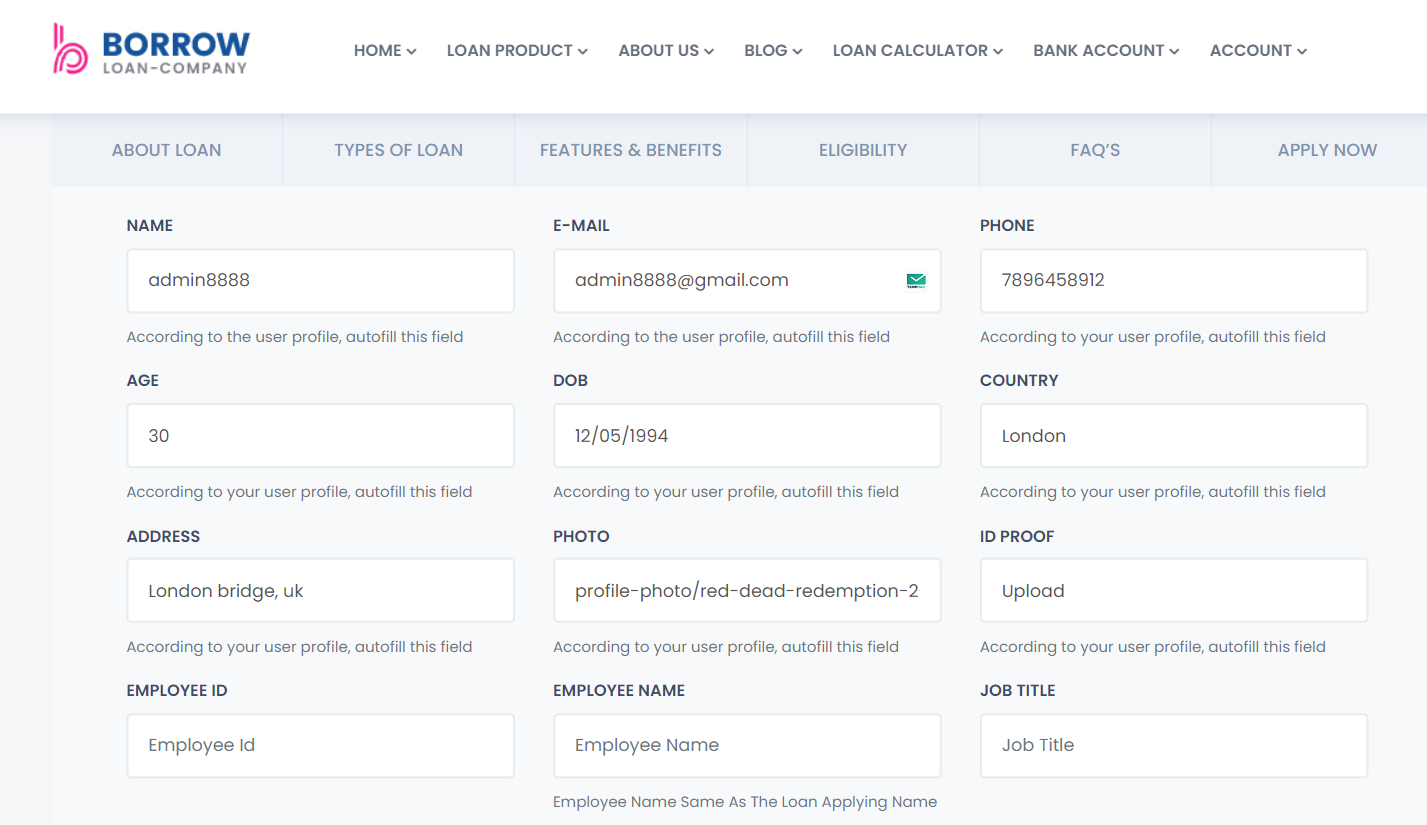


**Login and Sign-up page**









**REFERENCES**

The following materials were refered to develop the project Predict Medical Insurance :-

1.Software Engineering A Practitioner’s Approach 7th Edition

By Roger.Pressman 2.Database System Concepts 7th Edition

By Abraham Silberschatz, Henry F.Korth

S.Sudarshan

3.Data Mining Concepts and Techniques 3rd Edition

By Jiawei Han Micheline Kamber

Jian pei 4.https://[www.javatpoint.com](http://www.javatpoint.com/) [5.h](http://www.admecindia.co.in/)tt[p://www.admecindia.co.in](http://www.admecindia.co.in/)

6. https://[www.telusko.com](http://www.telusko.com/)