A

PROJECT REPORT

**On**

“Bank App”

of

Submitted in partial fulfillment of their requirements for the award of the Three-Year Diploma in

**Branch Name(CS/IT)**

Under the supervision of

**Brijesh Mishra**

**(Sr. Manager - IT)**

**Softpro India Computer Technologies (P) Ltd.**

**Lucknow (UP)**

**Submitted By:-**

**Student Name**

**Submitted To:-**

**College Name**

**PREFACE**

*“Necessity is Mother of All Inventions”*

Summer training is an important part of the engineering curriculum. The Diploma course summer training helps a student in getting acquainted with the manner in which his/her knowledge is being practically used outside his/her institute and this is normally different from what he/she has learnt from books. Hence, when the student switches from the process of learning to that of implementing his/her knowledge, he/she finds an abrupt change. This is exactly why summer training session during the B.T.E curriculum becomes all the more important. Summer training is prescribed for the student of Technical College as a part of the four-year degree course of engineering by the AICET. We are required to undergo summer training for a period of 45 days after the completion of the 2nd year.

This training report describes in detail the training after the 3rd year session, which I completed at the ***Softpro India Computer Technologies (P) Ltd.*** This report also gives the information about the organization and it’s working along with the project undertaken in the training period.

The fundamental step used in **SDLC** process is based on the ISO 9001 guidelines. My aim was to follow the ISO guidelines and develop a perfect system.

The system development was organized into 5 major parts:

# Requirement Gathering

1. **Documentation/Design**
2. **Development**
3. **Coding**
4. **Testing**

**Acknowledgement**

I would like to express my deep and sincere gratitude to my supervisor **Brijesh Mishra, Sr. Manager - IT (**Softpro India Computer Technologies (P) Ltd.)***,*** who gave me his full support and encouraged me to work in an innovative and challenging project for Educational field. His wide knowledge and logical thinking gave me right direction all the time.

I am deeply grateful my project coordinator for his help and support provided at every step of the project. Last but not the least, I thank to all employees of **Softpro India Computer Technologies (P) Ltd.** for their support and co-operation.

**Student Name**

SOFTPRO INDIA



COMPUTER TECHNOLOGIES PVT.LTD.

(AN ISO 9001:2015 CERTIFIED COMPANY)

**SPI/2021/VT-21/…..……..**

**Completion Certificate**

This is to certify that Mr. /Ms. **Student Name** of **Diploma (Branch Name)** from **College Name** (Institute/University) was working on the project entitled “**Bank App**” in Softpro India Computer Technologies Pvt. Ltd.. She/he was engaged with us during **15 August** to **30 September** for a period of **45 days.**

She/he has done an excellent job during his/her engagement with the Software Development & Testing Division of the company. She/he has completed his/her project during the training tenure. His/her performance has been good and satisfactory.

I would like to take this opportunity to express my appreciation to Mr./Ms. **Student Name** for his/her work and wish him/her all the very best for his/her future endeavors.

**Regards,**

**Ms. Yashi Asthana**

**CEO**

**Softpro India Computer Technologies (P) Ltd.**

**Lucknow (U.P.)**

**DECLARATION**

This is to certify that the project report entitled “**Bank App** ” is done by me is an authentic work carried out for the partial fulfillment of the requirements for the award of the Diploma in **“(Branch Name)”** under the guidance of **Brijesh Mishra**. The matter embodied in this project work has not been submitted earlier for award of any degree or diploma to the best of my knowledge and belief.

**Student Name**

**INDEX**

1. **Introduction………………………………………………………………………**
   1. **Overview of Organization**
   2. **Objectives**
   3. **Existing System Description**
   4. **Proposed System**
   5. **Tools Used**
2. **System Analysis…………………………………………………………………**
   1. **Objective**
   2. **SDLC Phases**
      1. **Feasibility Study**
      2. **Report Approval**
      3. **System Analysis**
      4. **System Design**
      5. **Coding**
      6. **Testing**
      7. **Implementation**
      8. **Maintenance**
   3. **Process Description**
   4. **Project Model Used**
   5. **ER-Diagram**
   6. **Data Flow Diagram**
3. **Software Requirement Specification………….………………………………**
   1. **Hardware Requirement**
   2. **Software Requirement**
   3. **Support Maintenance**
4. **System Design Approach…………………………………………………………**
   1. **Top-Down Designing**
   2. **Bottom –Up Designing**
   3. **Following Approach**
5. **Low Level Design…………………………………………………………………………**
6. **Data Modeling……………………………………………………………………**
   1. **List of Tables**
   2. **Structure of Tables**
7. **Testing………………………………………………………………………………**
8. **Input-Output Forms………………………………………………………………**
   1. **Modularization Details**
   2. **User Screenshots**
   3. **Admin Screenshots**
9. **Future Scope………………………………………………………………………**
10. **Conclusion……………………………………………………………………….**

**Introduction**

Bank App is an android based basic banking operation with its full functionalities. In this android app the bank customer can perform basic banking operations like deposit money, withdraw money, balance enquiry.

**Problem Definition**

In this section we shall discuss the limitation and drawback of the existing system that forced us to take up this project. Really that work was very typical to manage the daily errors free records and adding or removing any node from server. This problem produces a need to change the existing system. Some of these shortcomings are being discussed below: -

* **Low Functionality**

With the existing system, the biggest problem was the low functionality. The problem faced hampered the work.

* **Erroneous Input and Output**

In the existing system, humans performed all the tasks. As in the human tendency, error is also a possibility. Therefore, the inputs entered by the person who is working in the Company, in the registers may not be absolutely foolproof and may be erroneous. As a result of wrong input, the output reports etc. Will also be wrong which would in turn affect the performance.

* **Portability Problem**

System that existed previously was manual. As a result, the system was less portable. One has to carry the loads of many registers to take the data from one place to another. A big problem was that the system was less flexible and if we wanted to calculate yearly or monthly maintenance report or efficiency report, then it was a big headache.

* **Security-**

Security concerns were also one of the motives of the Company for the need of software. In the registers, the data is not secure as anybody can tamper with the data written in the registers. While in this software, just a password makes it absolutely secure from the reach of unauthorized persons.

* **Data Redundancy**

In the case of manual system, the registers are maintained in which, a lot of data is written.

* **Processing Speed**

In manual system maintaining a register and performing the necessary calculation has proved to be a troublesome job, which takes a lot of time and may affect the performance of the Company. But with this software we can have all the tasks performed in a fraction of second by a single click thus making the troublesome job much easier.

* **Manual Errors**

When a number of tough tasks are prepared by the humans like preparation of reports, performing long calculation then some human error is obvious due to a number of factors like mental strain, tiredness etc. But as we all know that computer never get tired irrespective of the amount of work it has to do. So, this software can nullify the probability of manual error that improve the performance.

* **Complexity in Work**

In manual system whenever a record is to be updated or to be deleted a lot of cutting and overwriting needs to be done on the registers that are concerned that are deleted or updated record, which makes the work very complex.

**Existing System Description**

The existing system bank customer perform banking operation manully. This was ineffective as down time was very high.

**Proposed System**

Softpro proposed to re-engineer the front-end of the workstation application to create a user interface specific to CSR needs for optimal call workflow. Softpro was initially a facilitator in discovery stage (Phase I) and was later selected as the system integrator. The iterative, time-boxed approach adopted by the Softpro to address the client’s requirements was as follows:

* **Analyze As-Is Process:**
  + A core working group was formed consisting of CCC Associates, Business analysts and IT personnel to go through the current processes and understanding problems.
  + Policies and technical issues were logged.
  + Prepare ‘as-is’ documentation.
* **Define To-Be Processes:**
  + Came up with the ‘to-be’ processes.
  + Went through regressive reviews by different business process owning departments.
  + Mapped ‘as-is’ with ‘to-be’.
  + Prepared User Interface Requirement Documents.
* **Develop Prototype:**
  + Build a prototype with the ‘to-be’ call workflows.
  + Went through regressive reviews by different business process owning departments for every screen.
* **Conducted time and motion study.**
* **Strategize ‘to-be’ architecture.**
* **Developed proof-of-concept**.

**System Analysis**

The complete understanding of software requirements is essential for the success of a software development effort. The requirements analysis task is a process of discovery, refinement modeling and specification. The software scope initially established by the system engineer and refined during software project planning in detail. Modules of the required data, information and control flow, and operational behavior and created. Alternative solutions are analyzed and allocated to various software elements.

Both developer and client take and active role in requirements analysis and specification. The client attempts to reformulate a sometimes-nebulous concept of software function and performance into concrete detail. The developer acts as an interrogator, consultant and problem solver.

Requirements analysis is a software engineering task that bridges the gap between system level software allocation and software design.

**Phases:**

System Development Life Cycle (SDLC) mainly consists of the following 7 phases which can be detailed: -

**Preliminary Investigation: -**

This is the first phase of the system development life cycle. In this phase we tend to find out the needs of the client –what exactly does the client want? Before the development of any system the important point is to know the needs, objectives and scope of the system.

**Feasibility Study**: -

Feasibility study is the second step of the system development life cycle. Things are always easy at the beginning in any software process. In fact, nothing is in feasible with unlimited time and resources. But it is not the fact. So, practically we have to do in limited resources in a restricted time margin. So, for the system to be feasible, following points we have to consider.

The feasibility study is conducted to check whether the candidate system is feasible. The system which is selected to be the best against the criteria is there after designed and developed. The feasibility study takes in to consideration, the risks involved in the project development beforehand. Therefore, in this phase we have to do feasibility study which is the test of the website according to its work ability, impact on the organization, ability to meet user need and effective use of resources. We do the feasibility study for website to analyze the risks, costs and benefits relating to economics, technology and user organization. There are several types of feasibility depending on the aspect they cover. Import of these includes:

**Technical Feasibility:**

This is an important outcome of preliminary investigation. It comprises of following questions: -

* Can the work of projected one with the current equipment, existing software and available man power resource?
* If Technology is required what are the possibilities that it can be developed?

**Economic Feasibility:**

It deals with question related to the economy. It comprises of the following questions: -

* Are there sufficient benefits in creating the system to make the cost acceptable?
* Are the costs of not creating the system so great that the project must be undertaken?

**Legal Feasibility:**

It deals with the question related to the legal issues. It comprises of the following questions:

* Contract Signing
* Software License agreement
  + - * Issues related to cyber laws.
      * Legal issues relating to the man power contract.

**Operational Feasibility:**

The operational feasibility consists of the following activity: -

* + - * Will the system be useful if it is developed &implemented?
      * Will there be resistance from employee?

**Social & Behavioral Feasibility:**

It deals with the various issues related to the human behavior like: -

* + - * Whether the user be able to adapt a new change or not?
      * Whether the ambiance we are providing suits the user or not?

**Request Approval: -**

Request approval is the third phase of system development lifecycle. Request approval is the phase in which all the requirements which would be provide in the system are stated. The request approval is a sort of agreement between the client and the company which is building this software. Both the parties should be mutually agreed on the stated requirements.

**System Analysis:** -

System analysis is the phase following the phase of the request approval. In this phase we tend to analyze the overall system which we have to build. System analysis is the crucial part in SDLC.

**System Design:** -

System design means the designing of the system. The System can be done in either of the following two ways: -

* Logical System Design
* Physical System Design

**Coding:** -

Coding is the phase in which a developer codes using any programming languages. Coding constitutes only 20% of the whole project and which is easier to write. The coding work is also done in the teams; development of the system is usually done under the modular programming style, which can be either top-down approach or bottom-up approach.

**Testing:** -

Testing is the phase in which the system that has been developed is tested. Testing comprises of the 60% of the overall development of the system. Testing of the system is important because testing aims to uncover the different errors in the system. There are various different testing techniques that can be used for the testing of the system.

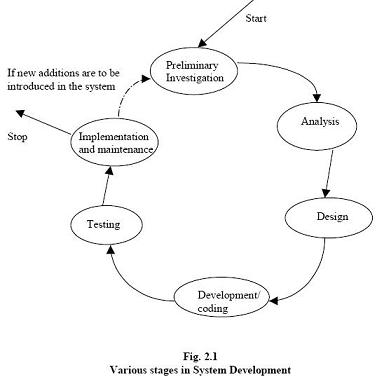
**Implementation: -**

Implementation process involved the installation of software on user’s side. Implementation process actually depends on type of a system &various. Opting for suitable conversion approach is a step implementation. The conversion processes are as follows: -

* Parallel Conversion
* Direct Conversion Approach
* Pilot Conversion Approach
* Phase in Conversion Approach

**Maintenance**: -

Merely developing the system is not important but also maintenance is important. The company that has built the system provides for some time free of cost maintenance to the client and after that period it is usually a paid service.

****

**Process Description**

Gantt charts mainly used to allocate resources to activities. The resources allocated to activities include staff, hardware, and software. Gantt charts (named after its developer Henry Gantt) are useful for resource planning. A Gantt chart is special type of bar chart where each bar represents an activity. The bars are drawn along a timeline. The length of each bar is proportional to the duration of the time planned for the corresponding activity.

Gantt chart is a project scheduling technique. Progress can be represented easily in a Gantt chart, by coloring each milestone when completed. The project will start in the month of June and end after one and half months at the beginning of August.

**Project Model Used**

**Iterative Enhancement Model**

* This model has the same phases as the waterfall model, but with fewer restrictions.
* Generally, the phases occur in the same order as in the waterfall model, but they may be conducted in several cycles.
* Useable product is released at the end of each cycle, with each release providing additional functionality. Customers and developers specify as many requirements as possible and prepare an SRS document. Developers and customers then prioritize these requirements. Developers implement the specified requirements in one or more cycles of design, implementation and test based on the defined priorities.

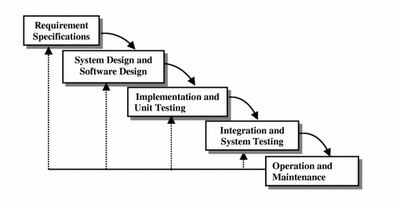
The procedure itself consists of the initialization step, the iteration step, and the Project Control List. The initialization step creates a base version of the system. The goal for this initial implementation is to create a product to which the user can react. It should offer a sampling of the key aspects of the problem and provide a solution that is simple enough to understand and implement easily. To guide the iteration process, a project control list is created that contains a record of all tasks that need to be performed. It includes such items as new features to be implemented and areas of redesign of the existing solution. The control list is constantly being revised as a result of the analysis phase.

The iteration involves the redesign and implementation of iteration is to be simple, straightforward, and modular, supporting redesign at that stage or as a task added to the project control list. The level of design detail is not dictated by the iterative approach. In a light-weight iterative project the code may represent the major source of [documentation](http://en.wikipedia.org/wiki/Software_documentation) of the system; however, in a critical iterative project a formal [Software Design Document](http://en.wikipedia.org/wiki/Software_Design_Document) may be used. The analysis of an iteration is based upon user feedback, and the program analysis facilities available. It involves analysis of the structure, modularity, [usability](http://en.wikipedia.org/wiki/Usability), reliability, efficiency, & achievement of goals. The project control list is modified in light of the analysis results.

### Phases:

Incremental development slices the system functionality into increments (portions). In each increment, a slice of functionality is delivered through cross-discipline work, from the requirements to the deployment. The unified process groups increments/iterations into phases: inception, elaboration, construction, and transition.

* Inception identifies project scope, requirements (functional and non-functional) and risks at a high level but in enough detail that work can be estimated.
* Elaboration delivers a working architecture that mitigates the top risks and fulfills the non-functional requirements.
* Construction incrementally fills-in the architecture with production-ready code produced from analysis, design, implementation, and testing of the functional requirements.
* Transition delivers the system into the production operating environment.

****

**Data Flow Diagram**

**Introduction:** -

DFD is an acronym for the word Data Flow Diagram. DFD is ppictorial representation of the system. DFD is a graphical representation of the ―flow of data through the information system. DFD are also used for the visualization of data processing (structured design). ADFD provides no information about the timings of the process, or about whether process will operate in parallel or sequence. DFD is an important technique for modeling a system’s high-level detail by showing how input data is transformed to output results through sequence of functional transformations. DFD reveal relationships among between the various components in a program or system. The strength of DFD lies in the fact that using few symbols we are able to express program design in an easier manner. A DFD can be used to represent the following: -

* External Entity sending and receiving data.
* Process that change the data.
* Flow of data within the system.
* Data Storage locations.

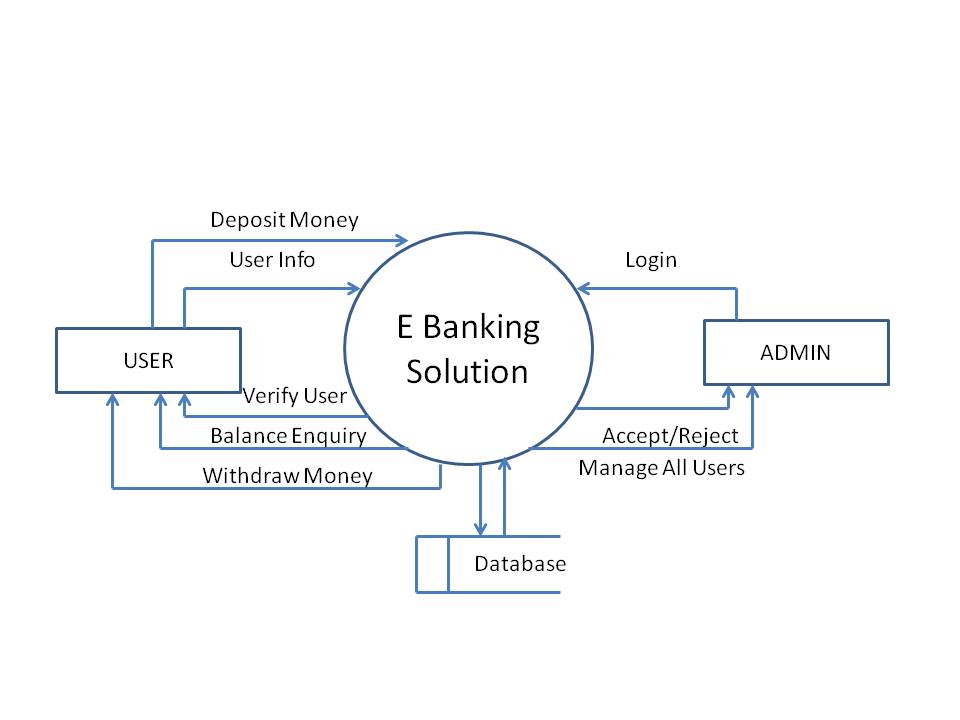
**Uses of DFD: -**

The main uses of data flow diagrams are as follows: -

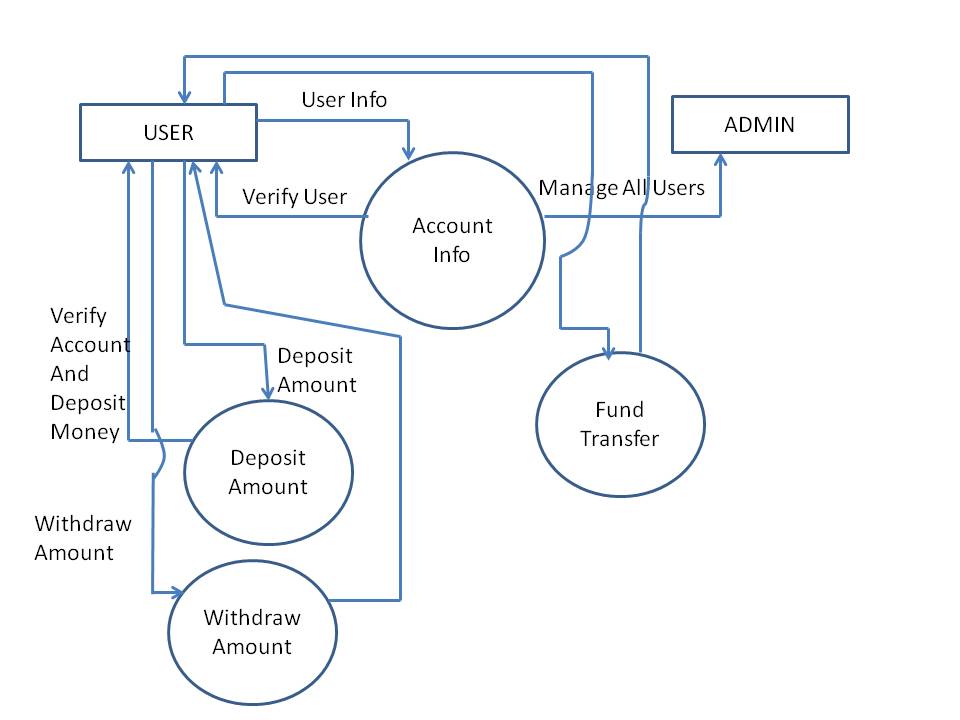
DFD is a method of choice for representation of showing of information through a system because of the following reasons:-

* DFDs are easier to understand by technical and non-technical audiences.
* DFDs can provide high level system overview, complete with boundaries and connections to other system.
* DFDs can provide a detailed representation of system components.

**0 Level DFD**

****

**1 Level DFD**

****

**SOFTWARE REQUIREMENT SPECIFICATION**

A requirements specification for a software system is a complete description of the behavior of a system to be developed and it includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements.

Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints). Requirements are a sub-field of software engineering that deals with the elicitation, analysis, specification, and validation of requirements for software.

The software requirement specification document enlists all necessary requirements for project development. To derive the requirements, we need to have clear and thorough understanding of the products to be developed. This is prepared after detailed communications with project team and the customer.

**Tools and Technology used:-**

1. Android Studio for App Development
2. MySql as network storage on server.

**Languages for programming:-**

1. Java
2. XML for UI development

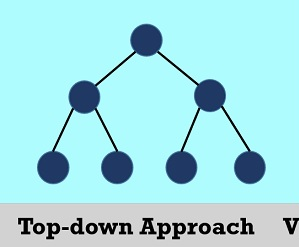
**SUPPORT AND MAINTENANCE: -**

One-year free support for rectifying system bugs including front end and beck end will be provided. During warranty period Software Engineers will be responsible for removing bugs and improving it. After one year support can be extended @ 20% of the total product deployment cost.

**SYSTEM DESIGN APPROACH**

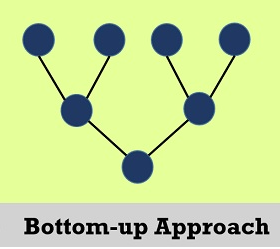
**Top – Down designing:**

The top - down designing approach started with major components of the system. It is a stepwise refinement which starts from an abstract design, in each steps the design is refined two or more concrete levels until we reach a level where no – more refinement is possible or not needed.



**Bottom – Up designing:**

In bottom – up designing the most basic and primitive components are designed first, and we proceed to higher level components. We work with layers of abstractions and abstraction are implemented until the stage is reached where the operations supported by the layer is complete.



**Approach we are following:**

In this project we are following **Mixed Approach** i.e. a combination of top – down and bottom – up. We are developing some of the components using top – down designing approach (e.g. the Web Pages) and some components in bottom – up designing approach (e.g. the middle tier classes).

**Modules in project:-**

**Modules:-**

1. Account Management
2. Deposit Money
3. Withdraw Money
4. Balance Enquiry
5. Login Management

**Low Level Design**

**Description:** Low Level Design creation is one of the most important activities in the development of any software product. The low-level design document gives the design of the actual software application. Low level design document is based on High Level Design document. It defines internal logic of every sub module. A good low-level design document will make the application very easy to develop by the developer. An effective design document results in very low efforts in developing a Software product.

Each project's low-level design document should provide a complete and detailed specification of the design for the software that will be developed in the project, including the classes, member and non-member functions, and associations between classes that are involved.

The low level design document should contain a listing of the declarations of all the classes, non-member-functions, and class member functions that will be defined during the subsequent implementation stage, along with the associations between those classes and any other details of those classes (such as member variables) that are firmly determined by the low level design stage. The low-level design document should also describe the classes, function signatures, associations, and any other appropriate details, which will be involved in testing and evaluating the project according to the evaluation plan defined in the project's requirements document.

**TESTING:**

Testing is the integral part of any System Development Life Cycle insufficient and interested application tends to crash and result in loss of economic and manpower investment besides user’s dissatisfaction and downfall of reputation.

“Software Testing can be looked upon as one among much process, an organization performs, and that provides the last opportunity to correct any flaws in the developed system. Software Testing includes selecting test data that have more probability of giving errors.” The first step in System testing is to develop the plan that all aspect of system. Complements, Correctness, Reliability and Maintainability.

Software is to be tested for the best quality assurance, an assurance that system meets the specification and requirement for its intended use and performance.

System Testing is the most useful practical process of executing the program with the implicit intention of finding errors that makes the program fail.

**Types of Testing:**

**Black Box (Functional) Testing:**

Testing against specification of system or components. Study it by examining its inputs and related outputs. Key is to devise inputs that have a higher likelihood of causing outputs that reveal the presence of defects. Use experience and knowledge of domain to identify such test cases. Failing this a systematic approach may be necessary. Equivalence partitioning is where the input to a program falls into a number of classes, e.g. positive numbers vs. negative numbers. Programs normally behave the same way for each member of a class. Partitions exist for both input and output. Partitions may be discrete or overlap. Invalid data (i.e. outside the normal partitions) is one or more partitions that should be tested.

Internal System design is not considered in this type of testing. Tests are based on requirements and functionality.

This type of test case design method focuses on the functional requirements of the software, ignoring the control structure of the program. Black box testing attempts to find errors in the following categories:

* Incorrect or missing functions.
* Interface errors.
* Errors in data structures or external database access.
* Performance errors.
* Initialization and termination errors.

**White Box (Structural) Testing:**

Testing based on knowledge of structure of component (e.g. by looking at source code). Advantage is that structure of code can be used to find out how many test cases need to be performed. Knowledge of the algorithm (examination of the code) can be used to identify the equivalence partitions. Path testing is where the tester aims to exercise every independent execution path through the component. All conditional statements tested for both true and false cases. If a unit has no control statements, there will be up to 2n possible paths through it. This demonstrates that it is much easier to test small program units than large ones. Flow graphs are a pictorial representation of the paths of control through a program (ignoring assignments, procedure calls and I/O statements). Use flow graph to design test cases that execute each path. Static tools may be used to make this easier in programs that have a complex branching structure. Tools support. Dynamic program analyzers instrument a program with additional code. Typically, this will count how many times each statement is executed. At end print out report showing which statements have and have not been executed. Problems with flow graph derived testing:

# Data complexity could not take into account.

# We cannot test all paths in combination.

# In really only possible at unit and module testing stages because beyond that complexity is too high.

This testing is based on knowledge of the internal logic of an application’s code. Also known as a Glass Box Testing. Internal software and code working should be known for this type of testing. Tests are based on coverage of code statements, branches, paths, conditions.

**Unit Testing:**

Unit testing concentrates on each unit of the software as implemented in the code. This is done to check syntax and logical errors in programs. At this stage, the test focuses on each module individually, assuring that it functions properly as a unit. In our case, we used extensive white-box testing at the unit testing stage.

A developer and his team typically do the unit testing do the unit testing is done in parallel with coding; it includes testing each function and procedure.

**Incremental Integration Testing:**

Bottom up approach for testing i.e. continuous testing of an application as new functionality is added; Application functionality and modules should be independent enough to test separately done by programmers or by testers.

**Integration Testing:**

Testing of integration modules to verify combined functionality after integration. Modules are typically code modules, individual applications, client and server and distributed systems.

**Functional Testing:**

This type of testing ignores the internal parts and focus on the output is as per requirement or not. Black box type testing geared to functionality requirements of an application.

**System Testing:**

Entire system is tested as per the requirements. Black box type test that is based on overall requirement specifications covers all combined parts of a system.

**End-to-End Testing:**

Similar to system testing, involves testing of a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with hardware, applications, or system if appropriate.

**Regression Testing:**

Testing the application as a whole for the modification in any module or functionality. Difficult to cover all the system in regression testing so typically automation tools are used for these testing types.

**Acceptance Testing:**

Normally this type of testing is done to verify if system meets the customer specified requirements. User or customers do this testing to determine whether to accept application.

**Performance Testing:**

Term often used interchangeably with “stress” and “load” testing, To check whether system meets performance requirements, Used different performance and load tools to do this.

**Alpha Testing:**

In house virtual user environment can be created for this type of testing. Testing is done at the end of development. Still minor design changes may be made as a result of such testing.

**Beta Testing:**

Testing typically done by end-users or others. This is final testing before releasing application for commercial purpose.

**SNAPSHOTS**

**­­­­**

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

**Coding:-**

Account\_Activity\_Create.xml

<**ScrollView android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@drawable/ban"  
 xmlns:android="http://schemas.android.com/apk/res/android"**>  
 <**LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"**>  
 <**TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Create New Acount..."  
 android:textSize="40dp"  
 android:gravity="center"  
 android:textColor="#00f"  
 android:textStyle="bold"  
 android:layout\_marginTop="30dp"**/>  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Enter Acount no."  
 android:id="@+id/et\_acno"  
 android:layout\_marginTop="30dp"  
 android:inputType="number"**/>  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Enter Name"  
 android:id="@+id/et\_name"  
 android:layout\_marginTop="30dp"**/>  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Enter Address"  
 android:id="@+id/et\_address"  
 android:layout\_marginTop="30dp"**/>  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Enter Contact no"  
 android:id="@+id/et\_contactno"  
 android:layout\_marginTop="30dp"  
 android:inputType="number"**/>  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Enter Email Address"  
 android:id="@+id/et\_emailaddress"  
 android:layout\_marginTop="30dp"** />  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Enter Pan Number"  
 android:id="@+id/et\_panno"  
 android:layout\_marginTop="30dp"**/>  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Enter Aadhar no"  
 android:id="@+id/et\_aadharno"  
 android:layout\_marginTop="30dp"  
 android:inputType="number"**/>  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Enter Openning Balance"  
 android:id="@+id/et\_balance"  
 android:layout\_marginTop="30dp"  
 android:inputType="number"**/>  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Enter Password"  
 android:id="@+id/et\_passwd"  
 android:layout\_marginTop="30dp"  
 android:inputType="textPassword"**/>  
 <**Button  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Create"  
 android:textSize="25dp"  
 android:onClick="create"  
 android:textAllCaps="false"  
 android:layout\_gravity="center"  
 android:textStyle="italic"  
 android:layout\_marginTop="50dp"  
 android:textColor="#00f"  
 android:layout\_marginBottom="50dp"**/>  
  
 </**LinearLayout**>  
</**ScrollView**>

Activity\_Deposit.xml

<**LinearLayout android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:background="#36ddec"  
 xmlns:android="http://schemas.android.com/apk/res/android"**>  
 <**ImageView  
 android:layout\_width="match\_parent"  
 android:layout\_height="250dp"  
 android:src="@drawable/deposit"  
 android:layout\_gravity="center"**/>  
 <**TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Deposit Amount"  
 android:textSize="30dp"  
 android:textColor="#090cc7"  
 android:textStyle="italic"  
 android:gravity="center"** />  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Enter the Amount to Deposit"  
 android:layout\_marginTop="40dp"  
 android:id="@+id/et\_depositamt"  
 android:inputType="number"**/>  
 <**Button  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Submit"  
 android:layout\_marginTop="20dp"  
 android:layout\_gravity="center"  
 android:textColor="#f11359"  
 android:textStyle="bold"  
 android:textSize="30dp"  
 android:textAllCaps="false"  
 android:onClick="deposit"**/>  
</**LinearLayout**>

Activity\_enquiry.xml

<**LinearLayout android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:background="#44e6d8"  
 xmlns:android="http://schemas.android.com/apk/res/android"**>  
 <**TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Balance Enquiry"  
 android:textColor="#00f"  
 android:textSize="40dp"  
 android:textStyle="bold"  
 android:layout\_marginTop="30dp"  
 android:gravity="center"** />  
 <**TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/tv\_balance"  
 android:text="Balance"  
 android:textSize="30dp"  
 android:textColor="#00f"  
 android:layout\_marginTop="50dp"  
 android:gravity="center"**/>  
</**LinearLayout**>

Activity\_main.xml

<**LinearLayout android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:background="@drawable/ban"  
  
 xmlns:android="http://schemas.android.com/apk/res/android"**>  
 <**ImageView  
 android:layout\_width="fill\_parent"  
 android:layout\_height="120dp"  
 android:src="@drawable/bank"  
 android:layout\_gravity="center"** />  
 <**TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Login Here.."  
 android:textSize="40dp"  
 android:textStyle="italic"  
 android:textColor="#9d0ce6"  
 android:gravity="center"  
 android:layout\_marginTop="2dp"** />  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/et\_userid"  
 android:hint="Enter Account no."  
 android:layout\_marginTop="20dp"  
 android:inputType="number"**/>  
  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/et\_password"  
 android:hint="Enter your Password"  
 android:layout\_marginTop="20dp"  
 android:inputType="textPassword"**/>  
  
 <**Spinner  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/spin\_op"  
 android:layout\_marginTop="20dp"**/>  
  
 <**LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:gravity="center"**>  
 <**Button  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Login"  
 android:textAllCaps="false"  
 android:onClick="login"  
 android:textColor="#fb0f1fcc"  
 android:layout\_marginTop="10dp"**/>  
 <**Button  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Create Account"  
 android:textAllCaps="false"  
 android:onClick="createaccount"  
 android:textColor="#fb0c1dde"  
 android:layout\_marginTop="10dp"**/>  
 </**LinearLayout**>  
</**LinearLayout**>

Activity\_withdraw.xml

<**LinearLayout android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:background="#36ddec"  
 xmlns:android="http://schemas.android.com/apk/res/android"**>  
 <**TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Withdraw Amount"  
 android:textColor="#00f"  
 android:textSize="40dp"  
 android:layout\_marginTop="30dp"  
 android:textStyle="bold"  
 android:gravity="center"**/>  
  
 <**EditText  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Enter Amount to Withdraw"  
 android:id="@+id/et\_withdrawamt"  
 android:layout\_marginTop="50dp"  
 android:inputType="number"**/>  
 <**Button  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Withdraw"  
 android:textAllCaps="false"  
 android:onClick="withdraw"  
 android:layout\_gravity="center"  
 android:layout\_marginTop="50dp"  
 android:textStyle="italic"  
 android:textSize="25dp"  
 android:textColor="#f50a41"**/>  
</**LinearLayout**>

CreateAcountActivity.java

**package** com.example.admin.bankapp;  
  
**import** android.content.Intent;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.EditText;  
**import** android.widget.Toast;  
  
**import** java.sql.Connection;  
**import** java.sql.PreparedStatement;  
**import** java.sql.SQLException;  
  
**public class** CreateAcountActivity **extends** AppCompatActivity {  
  
 EditText [] **ets**=**new** EditText[9];  
 **int** [] **ids**={R.id.***et\_acno***,  
 R.id.***et\_name***,  
 R.id.***et\_address***,  
 R.id.***et\_contactno***,  
 R.id.***et\_emailaddress***,  
 R.id.***et\_panno***,  
 R.id.***et\_aadharno***,  
 R.id.***et\_balance***,  
 R.id.***et\_passwd*** };  
 **int i**;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_create\_acount***);  
 **for**(**i**=0;**i**<**ets**.**length**;**i**++)  
 {  
 **ets**[**i**]=(EditText)findViewById(**ids**[**i**]);  
 }  
 }  
 **public void** create(View view)  
 {  
 **for**(**i**=0;**i**<**ets**.**length**;**i**++)  
 {  
 **if** (**ets**[**i**].getText().toString().isEmpty())  
 {  
 **ets**[**i**].setError(**"Empty"**);  
 **ets**[**i**].requestFocus();  
 **break**;  
 }  
 }  
 **if**(**i**==**ets**.**length**)  
 {  
 *//cod for create account* String [] values=**new** String[**ets**.**length**];  
 **for**(**i**=0;**i**<**ets**.**length**;**i**++)  
 {  
 values[**i**]=**ets**[**i**].getText().toString().trim();  
 }  
 DbManager dm=**new** DbManager();  
 Connection con=dm.getCon();  
 String query=**"insert into account values('"**+values[0]+**"','"**+values[1]+**"','"**+values[2]+**"','"**+values[3]+**"','"**+values[4]+**"','"**+values[5]+**"','"**+values[6]+**"','"**+values[7]+**"','"**+values[8]+**"')"**;  
 **try** {  
 PreparedStatement ps=con.prepareStatement(query);  
 ps.executeUpdate();  
 Toast.*makeText*(CreateAcountActivity.**this**,**"Account is created"**,Toast.***LENGTH\_SHORT***).show();  
 Intent I =**new** Intent(CreateAcountActivity.**this**,MainActivity.**class**);  
 startActivity(I);  
 finish();  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
}

DbManager.java

**package** com.example.admin.bankapp;  
  
**import** android.os.StrictMode;  
  
**import** java.sql.Connection;  
**import** java.sql.DriverManager;  
**import** java.sql.SQLException;  
  
*/\*\*  
 \* Created by admin on 31-Oct-21.  
 \*/***public class** DbManager  
{  
 **public** Connection getCon()  
 {  
 Connection con=**null**;  
 *//Perform Connection Online server to database Learn to Always.* StrictMode.ThreadPolicy policy=**new** StrictMode.ThreadPolicy.Builder().permitAll().build();  
 StrictMode.*setThreadPolicy*(policy);  
 **try** {  
 Class.*forName*(**"com.mysql.jdbc.Driver"**);  
 con= DriverManager.*getConnection*(**"jdbc:mysql://p3nlmysql51plsk.secureserver.net:3306/bankappdb"**,**"bankappdb"**,**"bankappdb@123"**);  
  
 } **catch** (ClassNotFoundException e) {  
 e.printStackTrace();  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 **return** con;  
 }  
  
}

DepositActivity.java

**package** com.example.admin.bankapp;  
  
**import** android.content.Intent;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.EditText;  
**import** android.widget.Toast;  
  
**import** java.sql.Connection;  
**import** java.sql.PreparedStatement;  
**import** java.sql.ResultSet;  
**import** java.sql.SQLException;  
  
**public class** DepositActivity **extends** AppCompatActivity {  
  
 EditText **et\_depositamt**;  
 String **acno**;  
 String **query**;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_deposit***);  
 **et\_depositamt**=(EditText)findViewById(R.id.***et\_depositamt***);  
 Intent I=getIntent();  
 **acno**=I.getStringExtra(**"acno"**);  
 }  
 **public void** deposit(View view)  
 {  
 **if**(**et\_depositamt**.getText().toString().isEmpty())  
 {  
 **et\_depositamt**.setError(**"Empty"**);  
 **et\_depositamt**.requestFocus();  
 }  
 **else** {  
 DbManager dm=**new** DbManager();  
 Connection con=dm.getCon();  
 **query**=**"select balance from account where acno='"**+**acno**+**"'"**;  
 **try** {  
 PreparedStatement ps=con.prepareStatement(**query**);  
 ResultSet rs=ps.executeQuery();  
 rs.next();  
 **int** balance=Integer.*parseInt*(rs.getString(**"balance"**));  
 **int** amt=Integer.*parseInt*(**et\_depositamt**.getText().toString());  
 balance=balance+amt;  
 **query**=**"update account set balance='"**+balance+**"'where acno='"**+**acno**+**"'"**;  
 ps=con.prepareStatement(**query**);  
 ps.executeUpdate();  
 Toast.*makeText*(DepositActivity.**this**,**"Balance is Created"**,Toast.***LENGTH\_SHORT***).show();  
 Intent I=**new** Intent(DepositActivity.**this**,MainActivity.**class**);  
 startActivity(I);  
 finish();  
 }  
 **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
}

EnquiryActivity.java

**package** com.example.admin.bankapp;  
  
**import** android.content.Intent;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.widget.TextView;  
  
**import** java.sql.Connection;  
**import** java.sql.PreparedStatement;  
**import** java.sql.ResultSet;  
**import** java.sql.SQLException;  
  
**public class** EnquiryActivity **extends** AppCompatActivity {  
  
 TextView **tv\_balance**;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_enquiry***);  
 Intent I=getIntent();  
 String acno=I.getStringExtra(**"acno"**);  
 **tv\_balance**=(TextView)findViewById(R.id.***tv\_balance***);  
 DbManager dm=**new** DbManager();  
 Connection con=dm.getCon();  
 String query=**"select balance from account where acno='"**+acno+**"'"**;  
 **try** {  
 PreparedStatement ps=con.prepareStatement(query);  
 ResultSet rs=ps.executeQuery();  
 rs.next();  
 **int** balance=Integer.*parseInt*(rs.getString(**"balance"**));  
 **tv\_balance**.setText(**"Your balance="**+balance);  
 } **catch** (SQLException e)  
 {  
 e.printStackTrace();  
 }  
 }  
}

MainActivity.java

**package** com.example.admin.bankapp;  
  
**import** android.content.Intent;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.EditText;  
**import** android.widget.ImageView;  
**import** android.widget.Spinner;  
**import** android.widget.Toast;  
  
**import** java.sql.Connection;  
**import** java.sql.PreparedStatement;  
**import** java.sql.ResultSet;  
**import** java.sql.SQLException;  
  
**public class** MainActivity **extends** AppCompatActivity {  
  
 EditText **et\_userid**,**et\_password**;  
 Spinner **spin\_op**;  
 String[] **op**= {**"select Operation"**,**"Deposit"**,**"Withdraw"**,**"Enquiry"**};  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 **et\_userid**=(EditText)findViewById(R.id.***et\_userid***);  
 **et\_password**=(EditText)findViewById(R.id.***et\_password***);  
 **spin\_op**=(Spinner)findViewById(R.id.***spin\_op***);  
 ArrayAdapter Ad=**new** ArrayAdapter(**this**,android.R.layout.***simple\_dropdown\_item\_1line***,**op**);  
 **spin\_op**.setAdapter(Ad);  
 }  
 **public void** createaccount(View view)  
 {  
 Intent I=**new** Intent(MainActivity.**this**,CreateAcountActivity.**class**);  
 startActivity(I);  
 finish();  
 }  
 **public void** login(View view)  
 {  
 **if**(**et\_userid**.getText().toString().isEmpty())  
 {  
 **et\_userid**.setError(**"Empty"**);  
 **et\_userid**.requestFocus();  
 }  
 **else** {  
 **if**(**et\_password**.getText().toString().isEmpty())  
 {  
 **et\_password**.setError(**"Empty"**);  
 **et\_password**.requestFocus();  
 }  
 **else** {  
 *//Here we write the code for login;* String userid=**et\_userid**.getText().toString().trim();  
 String password=**et\_password**.getText().toString().trim();  
 DbManager dm=**new** DbManager();  
 Connection con=dm.getCon();  
 String query=**"select password from account where acno='"**+userid+**"'"**;  
 **try** {  
 PreparedStatement ps=con.prepareStatement(query);  
 ResultSet rs=ps.executeQuery();  
 **if**(rs.next())  
 {  
 *//Account is exist* String respassword=rs.getString(**"password"**);  
 **if**(password.equals(respassword))  
 {  
 *//Toast.makeText(MainActivity.this,"Valid user",Toast.LENGTH\_SHORT).show();* String operation=**spin\_op**.getSelectedItem().toString();  
 **if**(operation.equals(**"Deposit"**))  
 {  
 *// Toast.makeText(MainActivity.this,"Diposit",Toast.LENGTH\_SHORT).show();* Intent I =**new** Intent(MainActivity.**this**,DepositActivity.**class**);  
 I.putExtra(**"acno"**,userid);  
 startActivity(I);  
 finish();  
 }  
 **else if**(operation.equals(**"Withdraw"**))  
 {  
 *//Toast.makeText(MainActivity.this,"Withdraw",Toast.LENGTH\_SHORT).show();* Intent I=**new** Intent(MainActivity.**this**,WithdrawActivity.**class**);  
 I.putExtra(**"acno"**,userid);  
 startActivity(I);  
 finish();  
 }  
 **else if**(operation.equals(**"Enquiry"**))  
 {  
 *//Toast.makeText(MainActivity.this,"Enquiry",Toast.LENGTH\_SHORT).show();* Intent I=**new** Intent(MainActivity.**this**,EnquiryActivity.**class**);  
 I.putExtra(**"acno"**,userid);  
 startActivity(I);  
 finish();  
 }  
 **else** {  
 Toast.*makeText*(MainActivity.**this**,**"Select Valid Operation"**,Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
 **else** {  
 Toast.*makeText*(MainActivity.**this**,**"Invalid user"**,Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
 **else** {  
 Toast.*makeText*(MainActivity.**this**,**"Account does not exist."**,Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
 **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
}

WithdrawActivity.java

**package** com.example.admin.bankapp;  
  
**import** android.content.Intent;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.EditText;  
**import** android.widget.Toast;  
  
**import** java.sql.Connection;  
**import** java.sql.PreparedStatement;  
**import** java.sql.ResultSet;  
**import** java.sql.SQLException;  
  
**public class** WithdrawActivity **extends** AppCompatActivity {  
  
 EditText **et\_withdrawamt**;  
 Intent **I**;  
 String **acno**;  
 String **query**;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_withdraw***);  
 **et\_withdrawamt**=(EditText)findViewById(R.id.***et\_withdrawamt***);  
 **I**=getIntent();  
 **acno**=**I**.getStringExtra(**"acno"**);  
  
 }  
 **public void** withdraw(View view)  
 {  
 **if**(**et\_withdrawamt**.getText().toString().isEmpty())  
 {  
 **et\_withdrawamt**.setError(**"Empty"**);  
 **et\_withdrawamt**.requestFocus();  
 }  
 **else** {  
 *//code for withdraw amount* DbManager dm=**new** DbManager();  
 Connection con=dm.getCon();  
 **query**=**"select balance from account where acno='"**+**acno**+**"'"**;  
 **try** {  
 PreparedStatement ps=con.prepareStatement(**query**);  
 ResultSet rs=ps.executeQuery();  
 rs.next();  
 **int** balance=Integer.*parseInt*(rs.getString(**"balance"**));  
 **int** amt=Integer.*parseInt*(**et\_withdrawamt**.getText().toString().trim());  
 **if**(balance<amt)  
 {  
 Toast.*makeText*(WithdrawActivity.**this**,**"Insuffienct balance"**,Toast.***LENGTH\_SHORT***).show();  
 **I**=**new** Intent(WithdrawActivity.**this**,MainActivity.**class**);  
 startActivity(**I**);  
 finish();  
 }  
 **else** {  
 balance=balance-amt;  
 **query**=**"update account set balance='"**+balance+**"' where acno='"**+**acno**+**"'"**;  
 ps=con.prepareStatement(**query**);  
 ps.executeUpdate();  
 Toast.*makeText*(WithdrawActivity.**this**,**"Amount is Dabited"**,Toast.***LENGTH\_SHORT***).show();  
 **I**=**new** Intent(WithdrawActivity.**this**,MainActivity.**class**);  
 startActivity(**I**);  
 finish();  
 }  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
}

AndroidManifast.xml

*<?***xml version="1.0" encoding="utf-8"***?>*<**manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.example.admin.bankapp"**>  
  
 <**uses-permission-sdk-23 android:name="android.permission.INTERNET"** />  
  
 <**application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:supportsRtl="true"  
 android:theme="@style/AppTheme"**>  
 <**activity android:name=".MainActivity"**>  
 <**intent-filter**>  
 <**action android:name="android.intent.action.MAIN"** />  
  
 <**category android:name="android.intent.category.LAUNCHER"** />  
 </**intent-filter**>  
 </**activity**>  
 <**activity android:name=".CreateAcountActivity"** />  
 <**activity android:name=".DepositActivity"** />  
 <**activity android:name=".WithdrawActivity"** />  
 <**activity android:name=".EnquiryActivity"**></**activity**>  
 </**application**>  
  
</**manifest**>

**FUTURE SCOPE**

The researcher has seen that the proposed solution can be expanded in the future. The following can be used to further enhance the system:

* Implement the application for different mobile platforms such as Windows and blackberry to cater for users who do not have Android mobile phones.