**Aadhaar Secure Travel Identity**

**A project report**

**Submitted in partial fulfillment**

**Of the requirements**

**For the award of the Degree of**

**Bachelor of Technology**

**In**

**COMPUTER SCIENCE AND ENGINEERING**

**By**

****

**Under the esteemed guidance of**

**Mr.D.Sasi Raja Sekhar**

**Head of the Department**

**Dept of CSE**

**Department of Computer Science and Engineering**

**(Affiliated to JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY)**

ACKNOWLEDGEMENT

It plunges me in exhilaration in taking privilege in expressing our heartfelt gratitude to **Mr.D.Sasi Raja Sekhar,** Head of CSE Department St.Mary’s College of Engineering and Technology, for providing every facility, constant supervision.

It gives us immense pleasure to acknowledge a debt of gratitude especially to our project incharge **Mr.Murlikrishana Rao,** department of CSE, St. Mary’s College of Engineering and Technology, for their constant encouragement, suggestions, constant supervision, and abundant support throughout our project.

We remain thankful to **Mr.Anil** external guide, **CMTES** for their invaluable support, Co-operation, and encouragement in completing our project successfully.

Thanks to all the teaching and non-teaching staff of St. Mary’s College of Engineering and Technology, for their support and also to our Team-mates for their valuable Co-operation.

**CONTENTS**

**S.NO. Page No.**

1. Project Abstract 1-2
2. Organization Profile 3-5
3. Cost Estimation 6-8
4. SRS 9-19
   1. Introduction
      1. Purpose
      2. Scope
      3. References
      4. Software Development Methodology
   2. System Analysis
      1. Study of the Existing System
      2. Problems in Existing System
      3. Proposed System
      4. Advantages of proposed system
      5. System Feature
      6. Requirement Specification
         1. Functional Requirements
         2. Non-Functional Requirements
   3. Feasibility Study
      1. Operational Feasibility
      2. Technical Feasibility
      3. Economical Feasibility
   4. System Requirements Study
      1. Software Requirements
      2. Hardware Requirements
   5. User Requirement Document(URD) 21-26
      1. Use-Case Diagram
      2. Activity Diagram
   6. System Design 27-29
      1. System Feature
      2. DFD
      3. Sequence Diagram 30-34
      4. Data Dictionary

4.6.4.1 Table Structure 35-37

4.6.4.2 ER Diagrams 38-40

* + 1. Class Diagram 41
    2. Deployment Diagram 42

1. Screen Shots 43-58 59-88
2. Coding 60-63
   1. Coding Standards & Guidelines 64-88
   2. Technology Trends
      1. Front End
      2. Back End
      3. Other Tools

6. Testing & Implementation 89-95

6.1 Test Case Description 89-91

* 1. Result Table 92-95

7. Conclusion 96-97

8. Bibliography 98-99

**PROJECT ABSTRACT**

**Project Title: Aadhaar Secure Travel Identity**

**Project Description:** This system is used to create a tool that manages the handling of passport and license using the unique identification associated with each individual. The application deals with allowing the citizens to register for a unique identity. The ID is supported with a pin. Citizen’s being issued passport or those have a passport is then associated with the UID. This helps the citizen to travel abroad without having the passport. The UID will provide access to the passport from the airport for the airline from the centralized server. The details and profile of the citizen with the photo can be viewed as part of security check.

The crime department can also use the application to trace or stop any person from travelling abroad. The airline gets a notification when the airport staff has access to the citizen’s passport. The crime department can stop or trace either using the UID or passport number. They could also pass the name of the person and the system can generate a list of photo previews of people having a passport.

The citizen uses the Aadhaar scheme to apply for license. The details of the citizen are picked from the registration database. The citizen is provided with the test details by the application. The details contain the location, date and time information. The test details are provided to the citizen on completion of the test. The license issue and denial is recorded.

**ORGANIZATION**

**PROFILE**

**CMTES SOLUTIONS**

**Profile:**

**CMTES** is an IT Solution Provider for a dynamic environment where business and technology strategies converge. Our approach focuses on new ways of business combining IT innovation and adoption while also leveraging an organization's current IT assets. We work with large global corporations and new generation technology companies - to build new products or services and to implement prudent business and technology strategies in today's environment.

CMTES Expertise includes:

1. Software Development Services
2. Engineering Services
3. Systems Integration
4. Customer Relationship Management
5. Supply Chain Management
6. Product Development
7. Electronic Commerce
8. Consulting
9. IT Outsourcing

We apply technology with innovation and responsibility to achieve two broad objectives:

* Effectively address the business issues our customers face today

1. Generate new opportunities that will help them stay ahead in the future

This approach rests on:

1. A strategy where we Architect, Integrate and Manage technology services and solutions — we call it AIM for success.
2. A robust offshore development methodology and reduced demand on customer resources
3. A focus on the use of reusable frameworks to provide cost and time benefits

We combine the best people, processes and technology to achieve excellent results — consistently. We offer customers the advantages of:   
  
**Speed:** We understand the importance of timing, of getting there before the competition. A rich portfolio of reusable, modular frameworks helps jump-start projects. Tried and tested methodology ensures that we follow a predictable, low-risk path to achieve results. Our track record is testimony to complex projects delivered within and even before schedule.

**Expertise:** Our teams combine cutting edge technology skills with rich domain expertise. What's equally important — we share a strong customer orientation that means we actually start by listening to the customer. We're focused on coming up with solutions that serve customer requirements today and anticipate future needs.  
  
**A Full Service Portfolio:** We offer customers the advantage of being able to Architect, Integrate and Manage technology services. This means that they can rely on one, fully accountable source instead of trying to integrate disparate multi-vendor solutions.

**Services:** CMTES is providing its services to Sain medicaments Pvt. Ltd, Grace drugs and pharmaceuticals pvt ltd alka drugs and pharmaceuticals pvt ltd to name just a few without rich experience and expertise in Information Technology we are in the best position to provide software solutions to distinct business requirements.

**COST ESTIMATION**

**EFFORT AND COST ESTIMATION**

**FUNCTION POINT MODEL**

It is based on the visible features of the system that are weighed accordingly to produce an overall score. The intent is to construct a measure of product size that can be available easily in the development process. It is based on the notion of function points regarding as a measure of functionality of the system. The starting point of the construction of the model is to determine the number of items occurring in the system.

The items are as follows:

**External inputs** are the inputs from the user that provide distinct application oriented data. Examples of such inputs are filenames and menu selections.

**External outputs** are directed to the user, they come in the form of various reports and messages.

**User inquiries** are interactive inputs requiring the response.

External files deal with all machine readable interfaces on other systems.

Internal files are the master files in the system.

These items are related differently according to their complexity that is given below in the following table.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Simple | Average | Complex |
| External Inputs | 3 | 4 | 6 |
| External Outputs | 4 | 5 | 7 |
| User Inquiries | 3 | 4 | 6 |
| External files | 7 | 10 | 15 |
| Internal files | 5 | 7 | 10 |

Here, first the Unadjusted Function Count (UFC) is determined using the formula

UFC = ∑itemi wi

In the second phase, refining the Function Point Count by including Technical Complexity Factor (TCF) and multiplying the value with UFC by using the formula determine Adjusted Function Point Count (FP):

**FP = UFC \* TCF**

Where TCF is calculated using the formula:

TCF = 0.65 + 0.1∑fi

Where fi specifies the detailed factors contributing to the overall notion of complexity.

The various factors are as follows-

* Reliable Backup and Recovery
* Distributed Functions
* Heavily used Configuration
* Operational Use
* Complex Interface
* Reusability
* Multiple sites
* Data Communications
* Performance
* Complex Processing
* Installation Ease
* Facilitate Change

Here each factor is rated on 0 to 5 scales with 0 being irrelevant and 5 standing for essential. If al the factors are irrelevant then the constant 0.65 is used otherwise the constant 1.35 is used.

**External inputs-**63(appno,ano,apwd,cname,caddr,dob,fname,cno,occ,ph,pin,status..)

**External outputs –** 5 (UID allocated, passport dispatched, license dispatched, transition status, change password)

**User inquiries –** 5(passport status, license status, view passport, view license,reports)

**External Files –** 1(photos)

**Internal Files –** 9(User’s info files)

Considering the Data Automation System, we assume the complexity of all items to be average.

So, UFC = (4\*63) + (5\*5) + (4\*5) + (10\*1) + (7\*9)

= 252 + 25 + 20 + 10 + 63

=370

TCF = 0.65 + 0.1(5 + 0 + 0 + 5 + 1 + 1 + 0 + 3 + 0 + 0 + 1 + 2)

= 2.45

Therefore FP = UFC\*TCF = 370\*2.45

= 906.5

**SYSTEM**

**REQUIREMENT**

**SPECIFICATION**

**(SRS)**

# 

# 4.1 Introduction

## 4.1.1 Purpose

To create a Tool that manages the handling of travel that includes passport, license and crime using the unique identification associated with each individual. Integrate Citizen, Aadhaar Dept, Crime, Passport & RTA to address travel issues and disputes.

## Intended Audience and Reading Suggestions

Document intended for Project Managers, Developers, End users and Quality Assurance engineers [Testing]. Suggested reading the document would be overall document description, begin with product based information gathering followed by Infrastructure requirement**.**

**4.1.2 Scope**

## Language Scope

**Language : Java [JSP, Servlets, JDBC]**

## Project Scope

* Citizen Registration, identification of each citizen using UID(Unique identification number)
* Secure passport application
* Passport application verification by the police authority in online mode.
* Passport Issue & Denial
* Provide crime department to integrate with the airport to stop someone from flying.
* Apply for License.
* Provide Test details and scenario - RTA.
* Provide test result.
* Issue or deny License.
* Change of address in one ID reflects the change in others.

## 4.1.3 References:

1) Java 1. Complete reference of Java - By Herbert Schildt

2. [www.w3schools.com](http://www.w3schools.com)

3. [www.java.sun.com](http://www.java.sun.com)

2) Database a) Database, Oracle

3) Html b) Html

4) UML The Unified Modeling Language User Guide

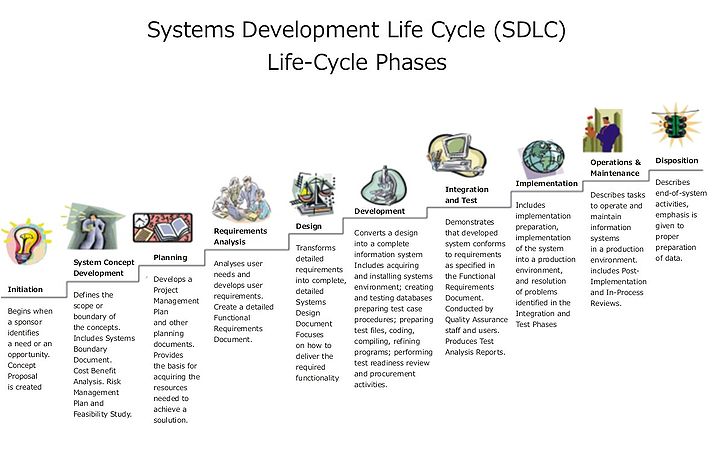
## Design and Implementation Constraints

The application requires a central server, similar to the one provided by the ISP. A well connected network of clients that connect to the server using the http protocol and a URL is required. Although the OS is not a dependent factor (JVM), a web server such as Tomcat has to be installed, configured and available throughout. A dedicated port number (8080) to which the incoming request and outgoing response has to be communicated should be assigned. The database server should be available for data access JDBC-ODBC drivers are required. The design of the application is addressed as follows,

* + - Aadhaar seva centers – Windows Forms ( Web forms)
    - Online / Server – Web forms

**Software Development Methodology:**

* **Project planning, feasibility study**: Establishes a high-level view of the intended project and determines its goals.
* **Systems analysis, requirements definition**: Refines project goals into defined functions and operation of the intended application. Analyzes end-user information needs.
* **Systems design**: Describes desired features and operations in detail, including screen layouts, business rules, process diagrams, pseudo code and other documentation.
* **Implementation**: The real code is written here.
* **Integration and testing**: Brings all the pieces together into a special testing environment, then checks for errors, bugs and interoperability.
* **Acceptance, installation, deployment**: The final stage of initial development, where the software is put into production and runs actual business.
* **Maintenance**: What happens during the rest of the software's life: changes, correction, additions, and moves to a different computing platform and more.



**SYSTEM ANALYSIS**

**Existing System:**

* + The citizen is identified by multiple identity cards.
  + There is no unique identity in India.
  + Passport has to be carried for travel abroad.
  + Crime department cannot talk to the private airlines to trace or stop travel of a citizen instantly.

**Disadvantage of Existing System:**

* The citizen is identified by multiple identity cards.
* There is no unique identity in India unlike the SSN in the USA.
* An individual can hold more than one passport.
* Possibility of passport being lost or damaged.
* Crime department communicates with the Airport Authority of India physically on stopping or tracing a citizen in travel.
* License can be applied multiple times, duplication possible, lost or damaged anytime.

**Proposed System:**

* A citizen is provided with a UID. The id is associated with a pin number.
* A physical verification is taken up by the surveyor on whose confirmation the ID is issued.
* A citizen holding the ID can only apply for passport or license.
* Based on the type of application the application is forwarded either to the Police department for verification or to the RTA for driving test status.
* Citizen has an online mode where he can check the status of each application.
* The crime department integrates with the airlines and identifies citizen who has a conditional travel.

**Advantage of Proposed System**

* For transaction related to government departments the ID and pin number should be quoted.
* The citizen does not have to approach agents for applications. Aadhaar seva centers would facilitate the application processing. The citizen has his application auto-filled when he visits the seva centers. No commission is involved.
* The UID doesn’t allow duplicate application for any type of card.
* The citizen can apply the next time only when he fails a verification or test.
* Address changes easily updated.

# System Features

1. Login & Security

The module allows only authorized users to use the application. The application comes with a default administrator account. This account has a user id “Admin” and password “Admin” by default. This user can change his password as and when necessary. The administrator maintains Aadhaar seva staffs that are authorized to handle the application and process transactions for the citizen.

1. Citizen Registration

The module allows the citizen to register for unique identity. The citizen provides his personal information, photo to the registration module. The module acknowledges by providing an application number. The citizen uses the application number to then check the status of the unique identity. The administrator is responsible to set the allocation status.

1. Process & Issue Unique Identity

The applications registered by the citizens are viewed by the administrator. When the administrator gets the physical verification report only then he confirms the issue of the unique identity. The module generates unique identification number for each application (citizen). The uid is also associated with a password which can be changed by the citizen.

1. Apply Passport

The module allows the citizen to register application for passport. The module auto fills the application with the information already available in the uid regn database. Only when there is a change of information the citizen should update the uid regn database and then apply for passport. For citizens already having passport and wanted change of information updating the uid regn database automatically updates the passport database.

1. Passport Processing

The module allows the administrator to verify the details of the application, set the status of the passport. If the application is accepted then the administrator provides the passport number and the system automatically updates the date of issue and expiry. This information is now available to the citizen as well as to the airport authority. The passport is a digital passport. The verification of the citizen is done by the crime/police dept.

1. Crime Control

The module allows the crime department to view citizen information either using the uid or the passport number. They can also provide the name of a person and have photos viewed for the citizens having identical names, highly useful when only the name of the person is known. The crime user can also set the status for a passport such as fly or no fly there by intimating the airport authority of flying restrictions for a citizen.

1. License Monitor

The module allows the citizen to apply for passport and have the details and the status of the test. The citizen will not have to submit any physical form nor have agents in between. The aadhaar scheme provides digital form and takes up citizen information from the registration database and dynamically fills the form. The test centers, their location are picked up based upon the pin code the citizen resides at. The RTA provides inputs on test schedules and test results to Aadhaar.

1. Reports
   * + Passport & License Applications
     + Application & Processing Status
     + Fly & No Fly List.

# FUNCTIONAL REQUIREMENTS

## User Interfaces

The application is provided with keyboard shortcuts with a facility to use the mouse to trigger the required actions. They act as shortcuts and provide an easy navigation within the software. Appropriate error handling is done using Exceptions in-order to isolate abnormal results or conditions. These messages are popped up to the user in the form of dynamic HTML or alerts.

## Hardware & Communication Interfaces

The application concentrates on the online and communicates over the internet/intranet. A well connected internet connection either using a modem or cable or Wi-Fi or any other form should exist. TCP/IP configured, http supported protocol configuration should exist. The client only requires a browser for communication. For Intranet requirement hubs/switches etc is a must. **Software Interfaces**

The incoming data to the product would be raw text data and images. The outgoing data would be the text and images. A database is maintained to store the text and URL information about the images. Ms-access/SQLServer is the database with a version of minimum 2003 as requirement. The server on the ISP requires tomcat web server. To execute or deploy the application JVM is required. A compatible browser is required to access the data from the client.

# NON FUNCTIONAL REQUIREMENTS

## Performance Requirements

Good band width, less congestion on the network. Identifying the shortest route to reach the destination would enhance performance.

## Safety Requirements

No harm is expected from the use of the product either to the OS or any data that resides on the client system.

## Product Security Requirements

The product is protected from un-authorized users from using it. The system allows only authenticated users to work on the application. The users of the system are Admin, passport officer, crime officer, RTA officer and the citizen.

**FEASIBILITY STUDY:**

Preliminary investigation examines project feasibility; the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All systems are feasible if they are given unlimited resources and infinite time.There are aspects in the feasibility study portion of the preliminary investigation:

* Operational feasibility
* Technical feasibility
* Economical feasibility

1. OPERATIONAL FEASIBILITY:

The application smart audit does not require additional manual involvement or labor towards maintenance of the system. Cost for training is minimized due to the user friendliness of the developed application.

Recurring expenditures on consumables and materials are minimized.

2. TECHNICAL FEASIBILITY:

Keeping in mind the existing system network, s/w

& H/w, already available the audit application generated in java provides an executable file that requires tomcat that provides compatibility from windows98 without having to load java software. No additional hardware or software is required which makes smart audit technically feasible.

3. ECONOMIC FEASIBILITY:

The system is economically feasible keeping in mind:

* Lesser investment towards training.
* One time investment towards development.
* Minimizing recurring expenditure towards training, facilities offered and consumables.
* The system as a whole is economically feasible over a period of time.

SYSTEM REQUIREMENT STUDY

**Software Requirement:**

Operating System : Windows 2000 or Higher

Service Pack : 2+

Platform : Java

Scripting : JSP

Backend : Ms-Access/Ms-Sql server

Web Server : Tomcat

**Hardware Requirement (Minimum)**

Processor : PIV

Ram : 512 Mb

Hard Disk : 10 GB Space

Monitor : VGA Color (256)

**USER REQUIREMENT DOCUMENT**

Use-case Diagram:

The **“user model view”** encompasses a problem and solution from the preservative of those individuals whose problem the solution addresses. The view presents the goals and objectives of the problem owners and their requirements of the solution. This view is composed of “use case diagrams”. These diagrams describe the functionality provided by a system to external interactors. These diagrams contain actors, use cases, and their relationships.

UID Reg

Login

Admin

Process UID

Passport Application

Passport Status

License App

RTA

License Issue/Denial

Citizen

Staff

Reports

Crime

Fly Status

AAI

**Activity Diagram**

**“Activity diagrams”** render the activities of a class participating in behavior. These diagrams describe the behavior of a class in response to internal processing rather than external events. Activity diagrams describe the processing activities within a class.

**Basic notation in Activity Diagram:**

**Initial node:** The filled in circle is the starting point of the diagram.

**Activity Final node:** The filled circle with a border is the ending point. An activity diagram can have zero or more activity final nodes.

**Activity:** The rounded rectangle represents activities that occur.

**Flow/Edge:** The arrow on the diagram.

**Fork:** A black bar with one flow going into it and several leaving it. This denotes the beginning of parallel activity.

**Join:** A black bar with several flows entering it and one leaving it. All flows going into the join must reach it before processing may continue. This denotes the end of parallel processing.

**Decision:** A diamond with one flow entering and several leaving.

**Login**

Accept uid and pwd

Verify

Error message

Admin

Citizen

Aadhaar Staff

Crimet

AAI

RTA

Citizen

NO

YES

**Citizen module**

Citizen

Error message

NO

YES

Accept uid and pwd

Apply License

View License

View Passport Status

Apply Passport

View UID Status

Apply for UID

**Aadhaar staff module**

Staff

Error message

NO

YES

Accept uid and pwd

Process License

Process Passport

Reports

Application for

Passport/license

**RTA module**

RTA

Error message

NO

YES

Accept uid and pwd

Reports

Declare Test Result

Schedule License

Application

**Crime Module**

Crime

Error message

NO

YES

Accept uid and pwd

Reports

Set Transit Status

passort Application

**AAI Module**

AAI

Error message

NO

YES

Accept uid and pwd

Reports

Check Transit

SYSTEM DESIGN

**INTRODUCTION**

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer’s goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word “Quality”. Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer’s view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

**DATA FLOW DIAGRAMS**

. The development of DFD’S is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level. The lop-level diagram is often called context diagram. It consists a A data flow diagram is graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams. Using two familiar notations Yourdon, Gane and Sarson notation develops the data flow diagrams. Each component in a DFD is labeled with a descriptive name. Process is further identified with a number that will be used for identification purposesingle process bit, which plays vital role in studying the current system. The process in the context level diagram is exploded into other process at the first level DFD.

The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for analyst to understand the process. Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical from, this lead to the modular design.

A DFD is also known as a “bubble Chart” has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

**DFD SYMBOLS:**

In the DFD, there are four symbols

1. A square defines a source(originator) or destination of system data
2. An arrow identifies data flow. It is the pipeline through which the information flows
3. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
4. An open rectangle is a data store, data at rest or a temporary repository of data

Process that transforms data flow.

Source or Destination of data

Data flow

Data Store

**CONSTRUCTING A DFD:**

Several rules of thumb are used in drawing DFD’S:

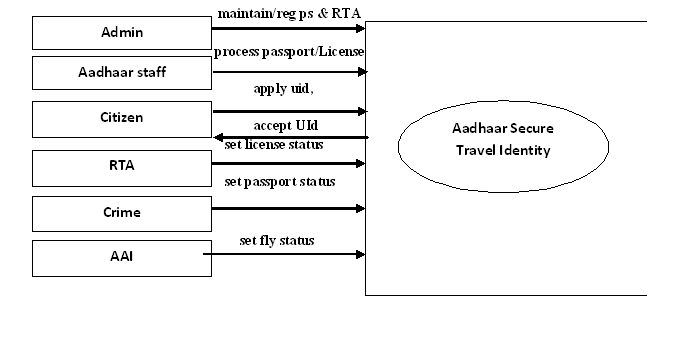
1. Process should be named and numbered for an easy reference. Each name should be representative of the process.
2. The direction of flow is from top to bottom and from left to right. Data traditionally flow from source to the destination although they may flow back to the source. One way to indicate this is to draw long flow line back to a source. An alternative way is to repeat the source symbol as a destination. Since it is used more than once in the DFD it is marked with a short diagonal.
3. When a process is exploded into lower level details, they are numbered.
4. The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each work capitalized

A DFD typically shows the minimum contents of data store. Each data store should contain all the data elements that flow in and out. Questionnaires should contain all the data elements that flow in and out. Missing interfaces redundancies and like is then accounted for often through interviews.

**SAILENT FEATURES OF DFD’S**

1. The DFD shows flow of data, not of control loops and decision are controlled considerations do not appear on a DFD.
2. The DFD does not indicate the time factor involved in any process whether the dataflow take place daily, weekly, monthly or yearly.
3. The sequence of events is not brought out on the DFD

**DATA FLOW DIAGRAM:**

****

SEQUENCE DIAGRAM:

UML sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modeling, which focuses on identifying the behavior within your system

A Sequence diagram is a two dimensional in nature. On the horizontal access, it shows the life of the object that it represents, while on the vertical axis, it shows the sequence of the creation or invocation of these objects.

1. LOGIN

\_

Registrations

\_

AADHAAR

\_

User

\_

1

:

\

User login request

\

\_

2

:

\

Prompt for uname

&

pwd

\

\_

3

:

\

Enter uname

&

pwd

\

\_

4

:

\

Verify

\

\_

5

:

\

Return status

(

Success

/

Failure

)

\

\_

6

:

\

Load modules

(

On success

)

\

\_

7

:

\

Load error page

(

On failure

)

\

2. UID GENERATION

A:Admin

T:Admin Op

C:citizen

Req UID Regn

Prompt citizen info

citizen Info

validate

save&update Uid db

Req Uid process

show appn,prompt appno

appno

populate pending appn's from UID DB

fetch details from UID DB

show appn Info

genarate uid,pin

save to uid db

status

status

apply status

3. APPLICATION

S:Staff Op

C:Citizen

P:pp Op

R:crime Op

req passport

Req Uid,ply status

prompt uid,status

issue status

save&update uid db

prompt uid

uid,pin

populate presonal profile from uid db

check issue status

fetch status from uid db

status

save&update db

req pp appn process

fetch appn's with issue status

show appn,prompt appno

appno

fetch profile Info

show Info

verification status report

update passport Info UId DB

status

Digital pp/deny

**4. SECURITY CHECK**

A:AAI Op

S:Security

officer

Req entry

prompt uid

uid,pin

populate personal profile from db

check flying status from uid db

entry allowed/disallow

5. LICENSE CHECK

C:citizen

S:Staff Op

R:RTA Op

req licence

prompt uid

uid, pin

populate personal profile from db

save&update uid db

req licence appn's

fetch appn's with non-issue status

show appn,prompt appno

appno

fetch profile Info

show Info

schedule test details

update licence Info in uid db

status

status

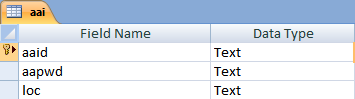
set test pass/fail Info

issue/deny license

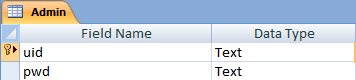
license details/fail

**DATA DICTIONARY**

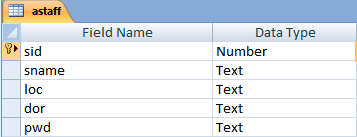
**Airport authority table**

****

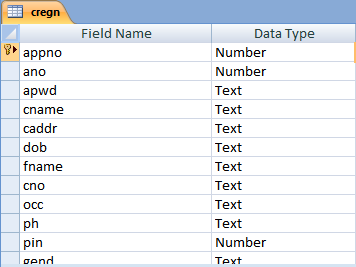
**Admin Table**

****

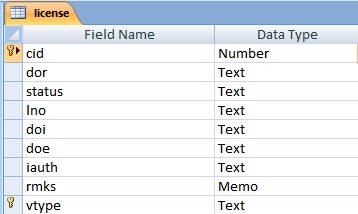
**Aadhaar Staff Table**

****

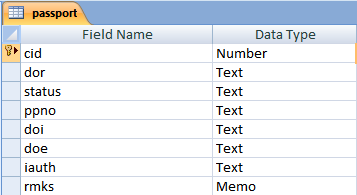
**Citizen Table**

****

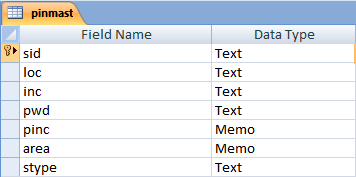
**License Table**

****

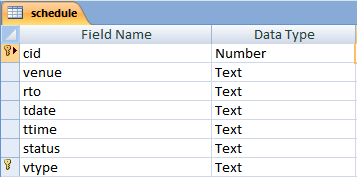
**Passport Table**

****

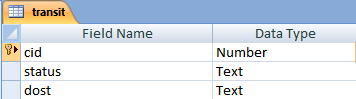
**Pin master Table**

****

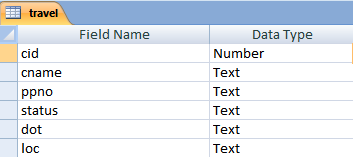
**License Status Table**

****

**Transition Status**

****

**Passport Travel Table**

****

**ER Diagram**

AID

APWDD

Maintain

ADMIN

PWD

UID

AAI

SID

SPWD

Register

Sname

ASTAFF

Maintain

Others

cname

Sname

SID

stype

pwd

area

Reg ps& RTA

cregen

cpwd

PINMAST

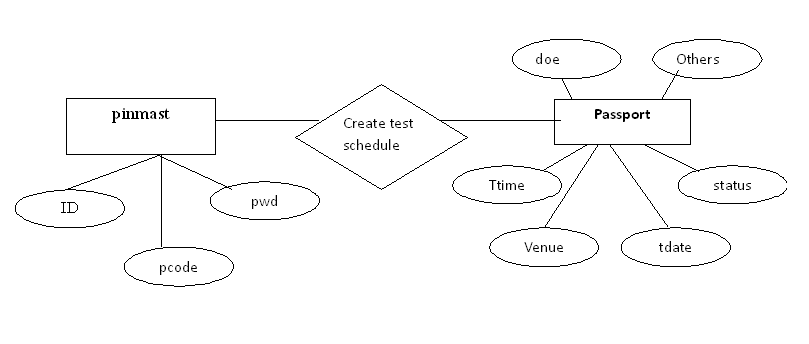
Others

cid

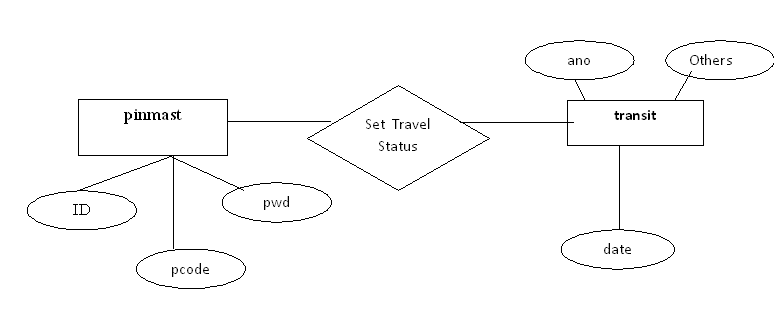
**Figure1**



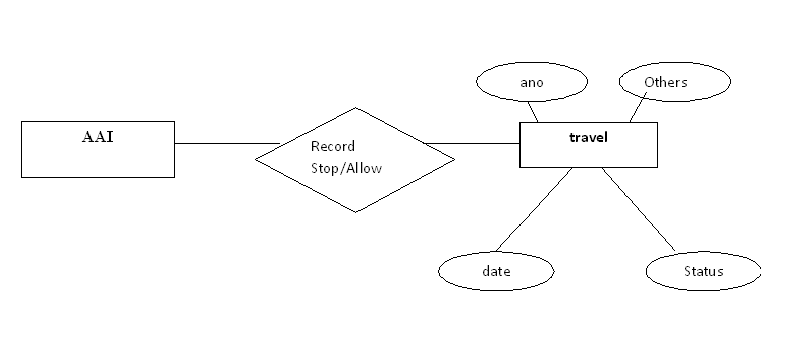
**Figure2**

****

**Figure3**



**Figure4**



**Figure5**

CLASS DIAGRAM:

*“***Class diagrams***”* describe the static structure of a system, or how it is declared rather than how it behaves. These diagrams contain classes and associations.



DEPLOYMENT DIAGRAM:

The “Environment model view*”* encompasses the structural and behavioral aspects of the domain in which a solution must be realized. This view is also known as the deployment or physical view. This view is composed of “deployment diagrams”. These diagrams describe the configuration of processing resources elements and the mapping of software implementation components onto them. These diagrams contain nodes, components and their relationships

Citizen

Admin

Staff

Crime

RTA

AAI

Middle Tier

Tomcat JSP

UIDDB

Apply UID

VIEW UID

Application

Process

Investigatio

n

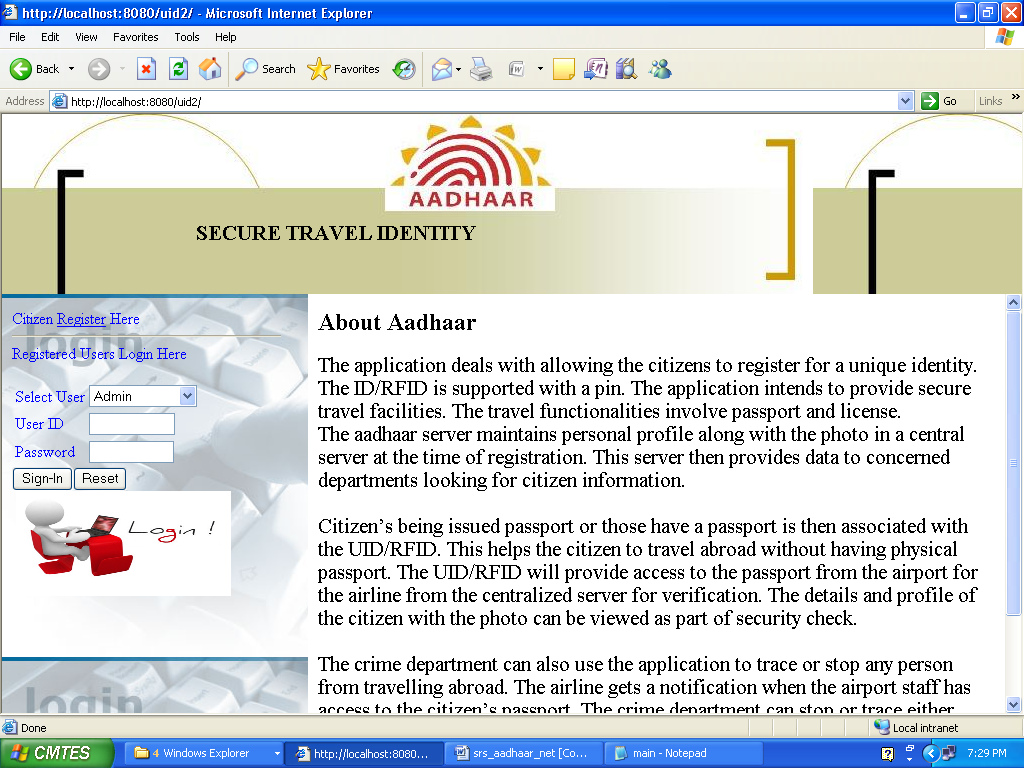
Schedule

test

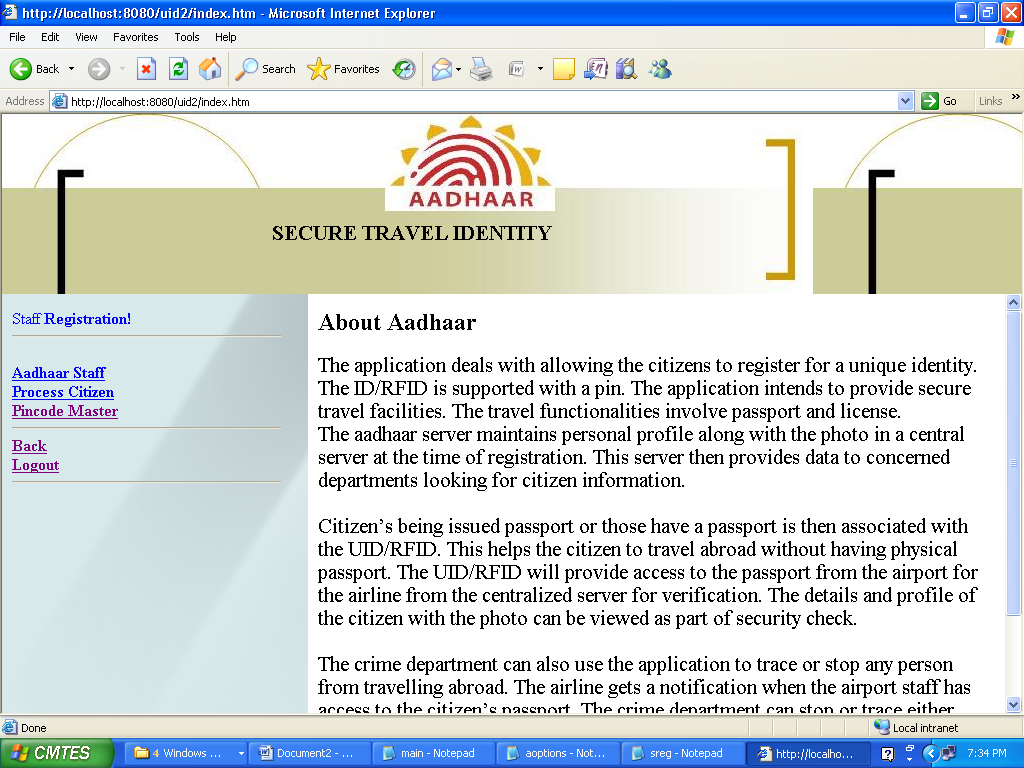
Flying

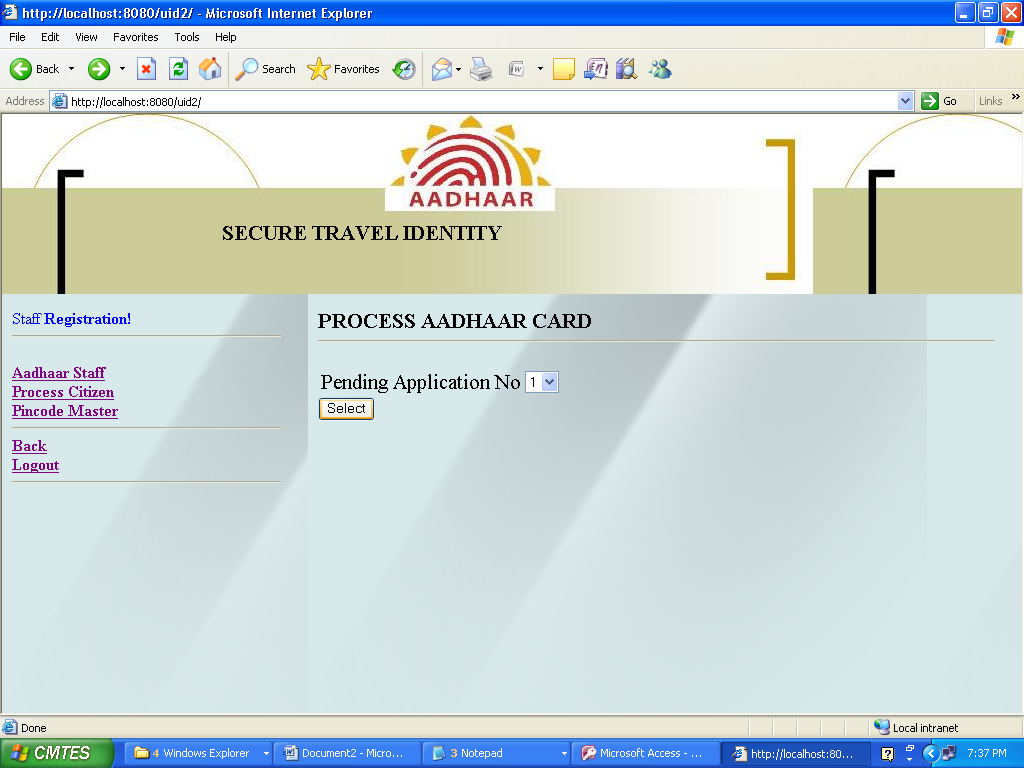
status

**SCREEN SHOTS:**





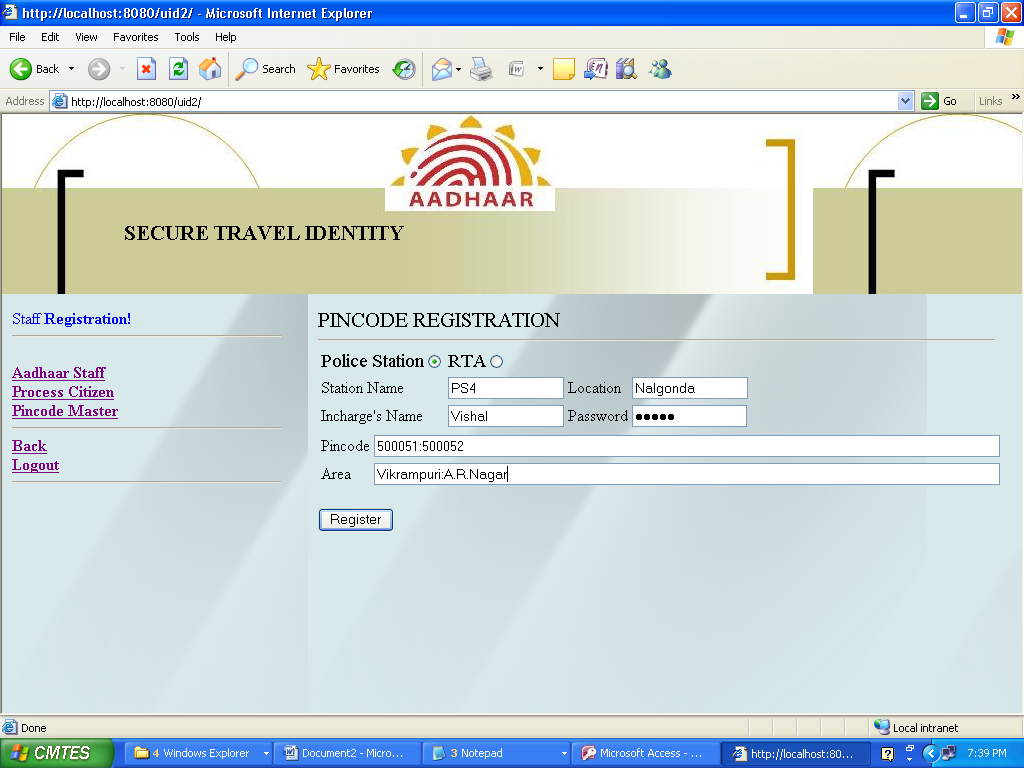


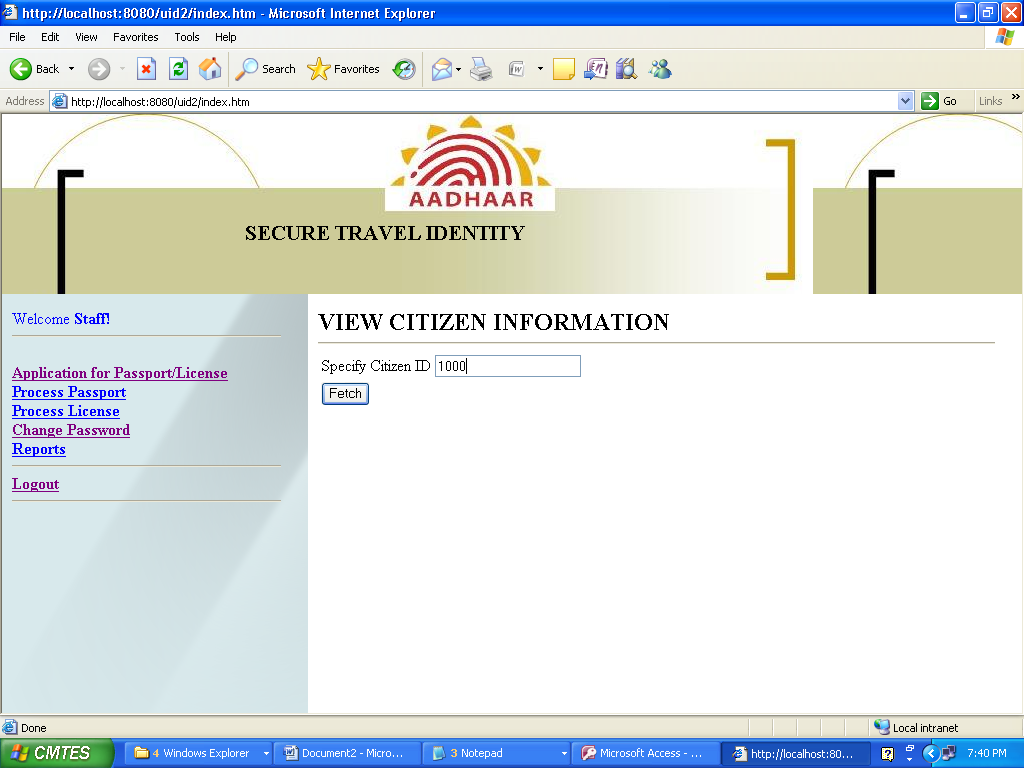








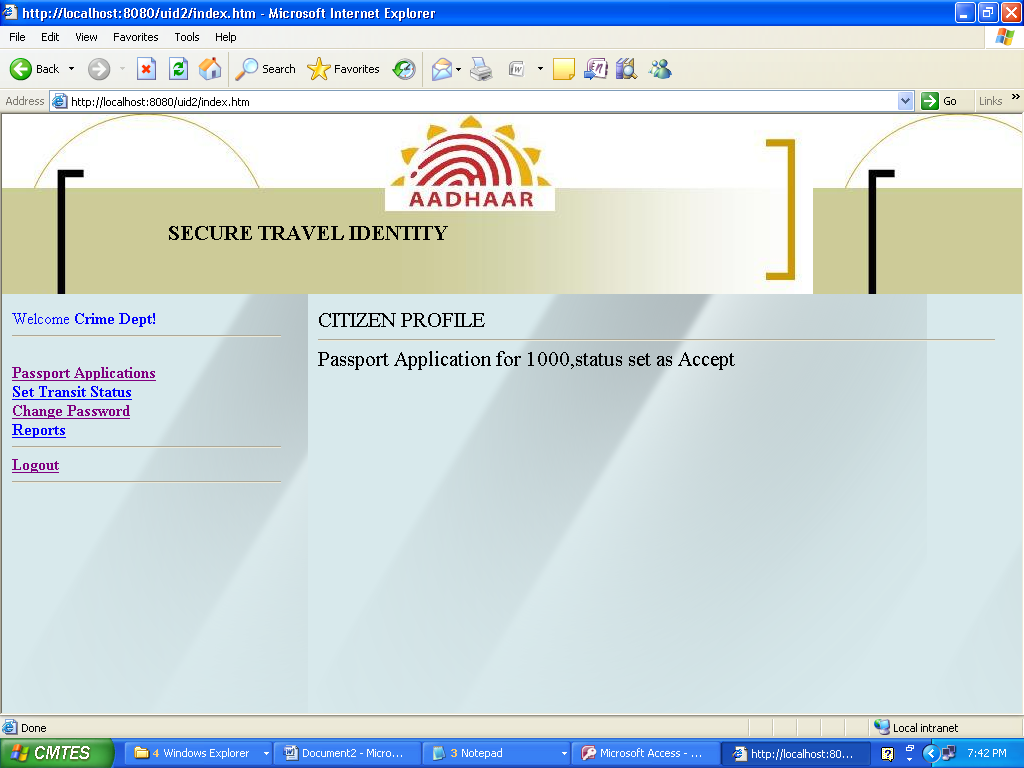


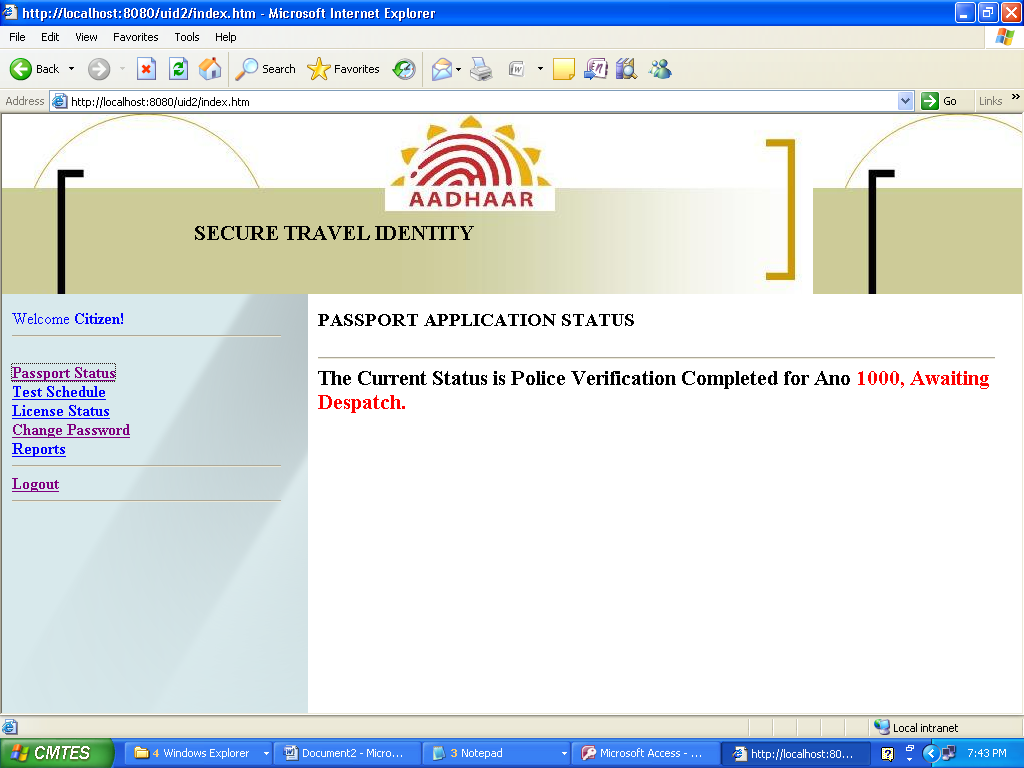


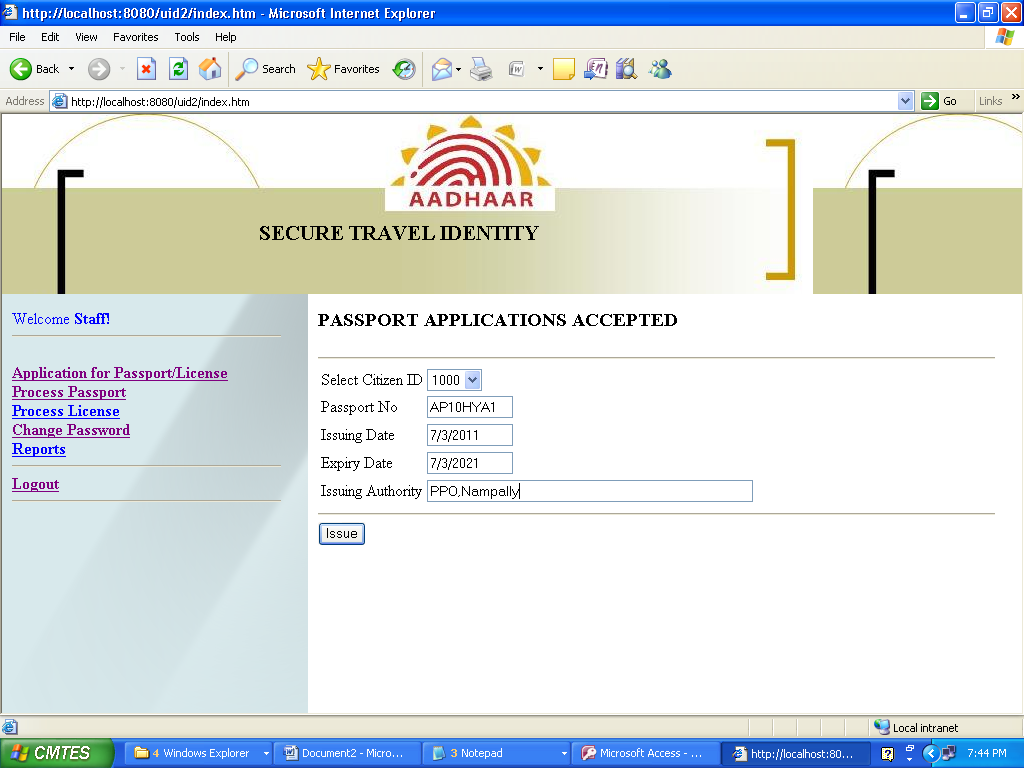


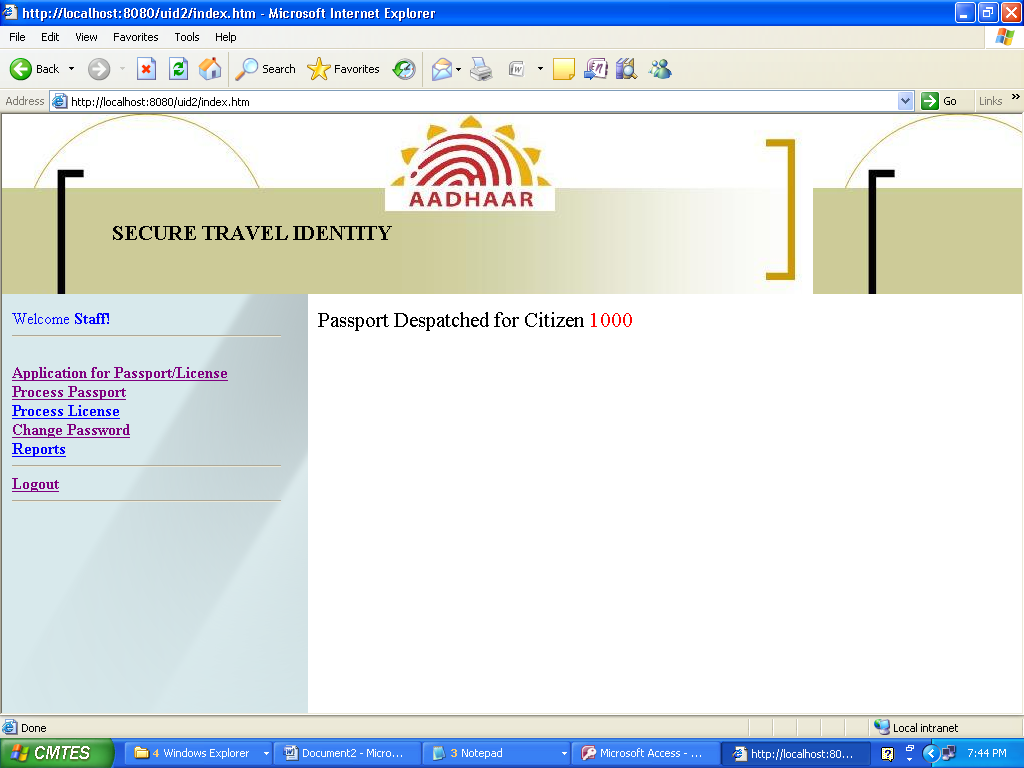






























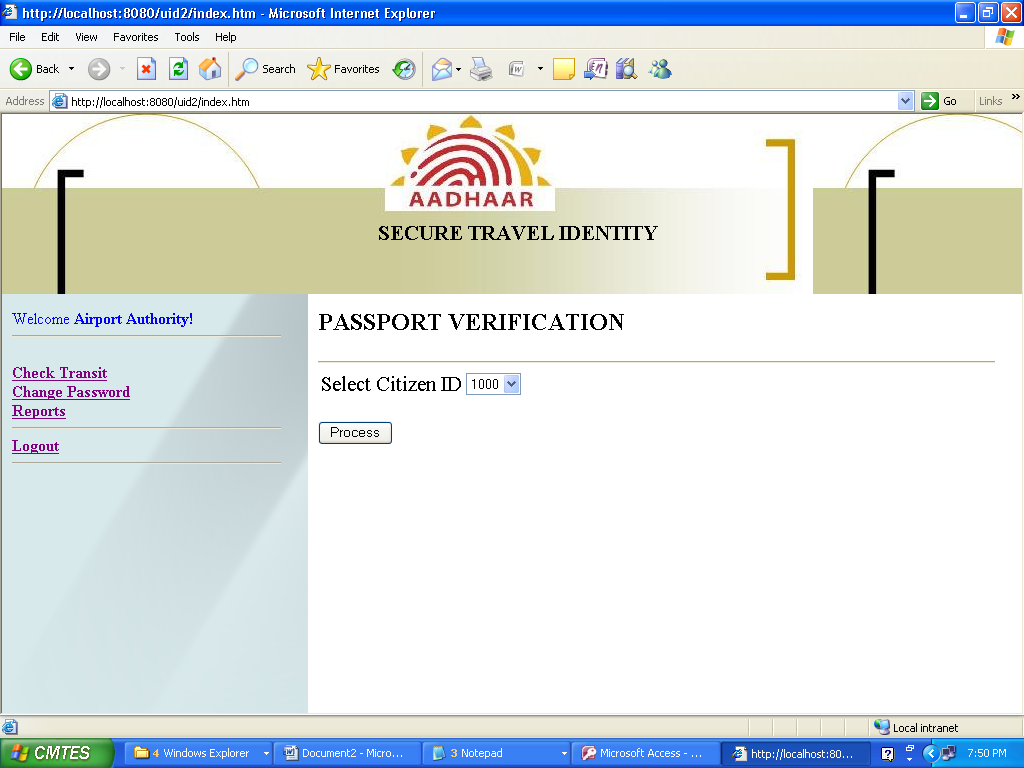


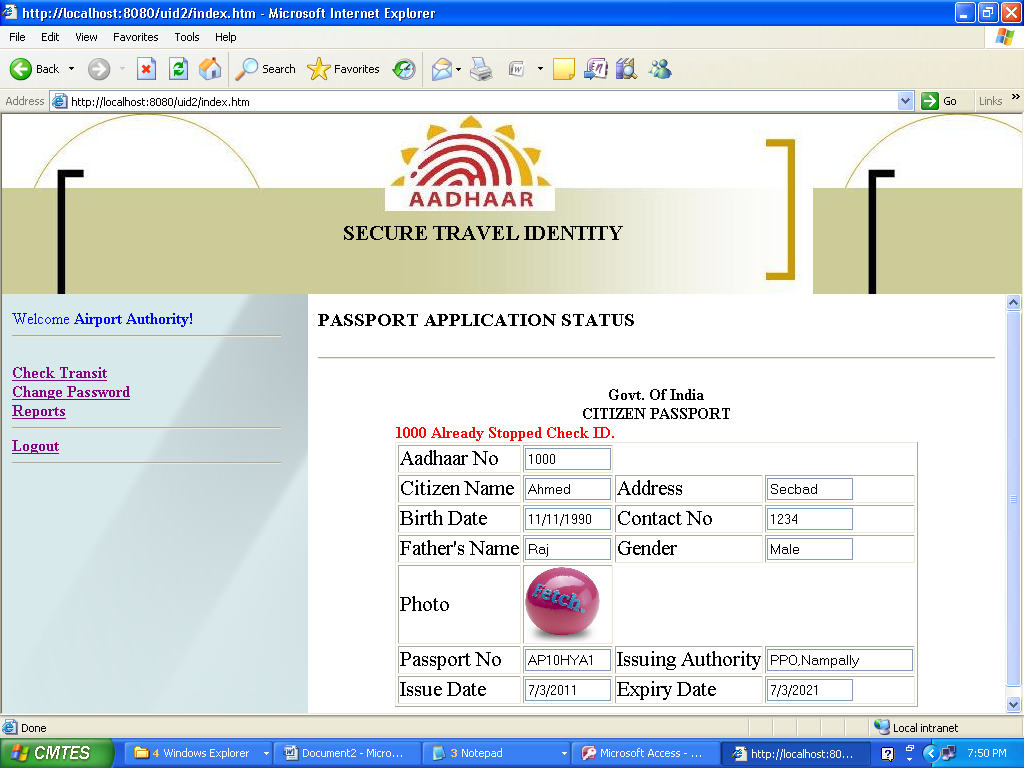












**CODING**

**Coding Standards:**

1. Introduction.. 2

2. Synchronization.. 2

3. Comments. 2

4. Transactions. 2

5. Servlets coding.. 3

6. DB Connection Pooling.. 4

**1.**    **Introduction**

This document describes a set of standards for developing programs based on J2EE architecture It is meant to be used not only by programmers specifically writing Java code, but also by programmers creating programs which automatically *generate* Java code.

**2.**    **Synchronization**

* Use the code or method synchronized whenever sees the method/code cannot be used asynchronously, when a particular value computed in a method/codes input to the next object call
* Usage of synchronous will affect the performance at the same time we must use in certain cases otherwise system will malfunction. Choose carefully and do not miss to use the synchronous key word wherever required.
* You can find the need (If you missed in design) if application is giving wrong output under load
* This happens because unsynchronized changes to shared data
* Use synchronize access to shared data
* But first try to avoid use of access to shared data if not possible then use synchronous keyword for synchronization.

**3.**    **Comments**

* Must use comments in the class, make sure you have used right comments, comments should not be confusing, must explain the underneath code implementation; please refer to i2labs Coding standards.

4**.**    **Transactions**

* Use Container Managed Transactions
* Associate each method with a single transaction
* Avoid Nested and Multiple Transactions
* In descriptor specify the methods associated with transactions
* Divide the transactions exception into Application level and System Level exceptions
* Catch Application-level exceptions from the business logic.
* System-level exceptions, such as runtime errors by exception handler/bean container. (Need to come with an approach) Usually role back
* Need to identify the methods which are marked for roll back in container, and Roll back the transaction for checked exec
* Once exception is thrown make the decision about continuing the transaction or not.
* Retry and continue the transaction. This might entail retrying portions of the transaction.

**5.**    **Servlets coding**

  **Points to remember**

1. Cache
2. Don’t over-synchronize
3. Limit object creation
4. Store bytes, not characters

**Note: Don’t create unnecessary Heaps for example,**

String buildForm(args) {

StringBuffer b = new StringBuffer();

b.append(....);

b.append(....);

b.append(....);

return b.toString();

This code creates three heaps

See the following--

void buildForm(StringBuffer b, args) {

b.append(....);

b.append(….)

b.append(....);

creates almost no heaps.

* Use long instead of java.util.Date
* Use several scalars instead of small array
* Use arguments and locals instead of member variables
* Performance improvement by using right code

See the following code

class HelloWorld extends HttpServlet

{

private String html = " ... ";

void doGet(request, response)

{

PrintWriter socket; socket = response.getWriter(); socket.print(html);

}

}

See the following code, which is approximately 3 times faster than the above code.

class HelloWorld extends HttpServlet

{

private String html = " ... ";

private byte[] b = html.getBytes();

void doGet(request, response)

{

OutputStream socket;

socket = response.getOutputStream();

socket.write(b);

}

}

* Converting chars bytes is *very slow*
* Always use OutputStream, never PrintWriter
* Store bytes, not characters

**6.**    **DB Connection Pooling**

* Use Application server connection pooling
* In case you are using your own connection pool, configure it to XA
* Explicit commit should not be used.
* Must to close every connection borrowed, use finally block at the end of try/catch block and close the connection

## Technology Trends

**Front End (JAVA)**

**INTRODUCTION:**

Java was conceived by James Gosling, Patrick Naughton, ChrisWarth, Ed Frank and Mike Sheridan at SUN Micro Systems Incorporation in 1991. It took 18 months to develop the first working version. This language was initially called “OAK”, but was renamed “JAVA” in 1995. Before the initial implementation of OAK in 1992 and the public announcement of Java in 1995, many more contributed to the design and evolution of the language.

**OVERVIEW OF JAVA**:

An Object Oriented Programming Language(OOPL) developed at Sun Microsystems. A Virtual Machine Run Time Environment that can be embedded in web browser (IE, NN). Java is a powerful but lean object oriented programming language. It has generated a lot of excitement because it makes it possible to program for Internet by creating applets, programs that can be embedded in web page.

The context of an applet is limited only by one’s imagination. For example, an applet can be an animation with sound, an interactive game or a ticker tape with constantly updated stock prices. Applets can be serious application like word processor or spreadsheet.

But Java is more than a programming language for writing applets. It is being used more and more for writing standalone applications as well. It is becoming so popular that many people believe it will become standard language for both general purpose and Internet programming. There are many buzzwords associated with Java, but because of its spectacular growth in popularity, a new buzzword has appeared ubiquitous. Indeed, all indications are that it will soon be everywhere.

***Java builds on the strength of C++. It has taken the best features of C++ and discarded the more problematic and error prone parts. To this lean core, it has added garbage collection (automatic memory management), multithreading (the capacity for one program to do more than one thing at a time), security capabilities. The result is simple, elegant, powerful and easy to use.***

Java is actually a platform consisting of three components:

* Java Programming Language.
* Java Library of Classes and Interfaces.
* Java Virtual Machine.

It also has a Standardized set of Packages (Class, Interfaces):

* Creating Graphical User Interfaces
* Controlling Multimedia Data
* Communicating over Network

The following sections will say more about these components:

#### FEATURES OF JAVA:

#### PORTABILITY:

One of the biggest advantages Java offers is that it is portable. An application written in Java will run on all the major platforms. Any computer with a Java based browser can run the applications or applets written in the Java Programming Language. A programmer no longer has to write one program to run on a Macintosh, another program to run on a windows machine, still another to run on a UNIX machine, and so on. In other words, with Java, developers write their programs only once. The virtual machine is what gives Java is cross platform capabilities.

Rather than being compiled into machine language, which is different for each operating systems and computer architecture, Java code is compiled into byte codes. With other languages, the program can understand. The problem is that other computers with different machine instruction set cannot understand that language. Java code, on the other hand is compiled into byte codes rather than a machine language. These byte codes go to the Java virtual machine, which executes them directly or translate them into the language that is understood by the machine running it. In Summary, these means that with the JDBC API extending Java, A programmer writing Java code can access all the major relational databases on any platform that supports the Java virtual machine.

**OBJECT\_ORIENTED**:

The Java programming language is object oriented, which makes program design focus on what you are dealing with rather than on how you are going to do something. This makes it more useful for programming in sophisticated projects because one can break the things down into understandable components. A big benefit is that these components can then be reused.

Object oriented languages use the paradigm of classes. In simplest term, a class includes both the data and the functions to operate on the data. You can create an instance of a class, also called an object, which will have all the data members and functionality of its class. Because of this, you can think of a class as being like template, with each object being a specific instance of a particular type of class.

The class paradigm allows one to encapsulate data so that specific data values are those using the data cannot see function implementation. Encapsulation makes it possible to make the changes in code without breaking other programs that use that code. If for example the implementation of a function is changed, the change is invisible to the programmer who invokes that function, and it does not affect his/her program, except hopefully to improve it. Java includes inheritance, or the ability to derive new classes from existing classes. The derived class, also called a subclass, inherits all the data and the functions of the existing class, referred to as the parent class. A subclass can add new data members to those inherited from the parent class. As far as methods are concerned, the subclass can reuse the inherited methods as it is, changed them, and/or add its own new methods.

**EASY**

In addition to being portable and object oriented, Java facilitates writing correct code. Programmers spend less time writing Java code and a lot less time debugging it. In fact, developers have reported slashing development time by as much as two thirds.

The following is a list of some of Java’s features that make it easier to write correct code:

**GARBAGE COLLECTION**:

Automatically takes care of allocating and reallocating memory, a huge potential source of errors. If an object is no longer being used (has no references to it), then it is automatically removed from memory. Dynamic binding ispossible and often very useful, but static binding with strict type checking is used when possible.

#### SIMPLICITY:

Makes Java easier to learn and use correctly. Java keep it simple by having just one way to do something instead of having several alternatives, as in some languages. Java also stays lean by not including multiple inheritance, which eliminates the errors and ambiguity that arise when you create a subclass that inherits from two or more classes. Java lets you add functionality to a class throws by the use of interfaces.

**JAVA INCLUDES A LIBRARY OF CLASSES AND INTERFACES:**

The Java platform includes an extensive class library so that programmers can use already existing classes, as it is, create subclasses to modify existing classes, or implement interfaces to augment the capabilities of classes.

Both classes and interfaces contain data members (fields) and functions (methods), but there are major differences. In a class, fields may be either variable or constant, and methods are fully implemented. In an interface, fields must be constants, and methods are just prototypes with no implementations. The prototypes give the method signature (the return type, the function name, and the number of parameters with the type for each parameter), but the programmer must supply implementations.

###### **To use an interface, a programmer defines a class, declares that it implements the Interface, and then implements all the methods in that interface as part of the class. These methods are implemented in a way that is appropriate for the class in which the methods are being used. Interfaces let one add functionality to a class and give a great deal of flexibility in doing it. In other words interfaces provide most of the advantages of multiple inheritances without its disadvantages.**

***A package is a collection of related Java classes and interfaces. The following list, though not complete, gives example of some Java packages and what they cover.***

* Java.lang: The basic classes. This package is so basic that it automatically is included in any Java program. It includes classes dealing with numeric, strings, objects, runtime, security, and threads.
* Java.io: Classes that manages reading data from input streams and writing data to the output streams.
* Java.util: Miscellaneous utility classes, including generic data structures, bit sets, time, date, the string manipulation, random number generation, system properties, notification and enumeration of data structures.
* Java.net: Classes for network support.
* Java.awt: Classes that manage user interface components such as windows, dialog boxes, buttons, and checkboxes, lists, menus, scrollbars, and text fields, the “AWT” stands for Abstract Window Toolkit.
* Java.awt.image: Classes for managing image data, including color models, dropping color flittering, setting pixel values, and grabbing snapshots.
* Java.applet: The Applet class, which provides the ability to write applets, this package also includes several interfaces that, connect an applet to its documents and to its document and to its document and to recourses for playing audio.
* Java.sql: The JDBC API, classes and interfaces that access databases and send SQL Statements.

The first three packages listed, java.lang, java.io and java.util form the foundation, they are basic classes and interfaces for general-purpose programming.

Java development kit version1.1 added some new packages, with JDBC being one of them. Other new packages include such thing as Remote Method Invocation, Security and Java Beans, the new API for creating reusable components.

In Java, packages serve as the foundation for building other packages, as discussed in the following section.

#### EXTENSIBILITY:

A big plus for Java is the fact it can be extended. It was purposely written to be lean with the emphasis on doing what it does very well, instead of trying to do everything from the beginning; it was return so that extending it is very easy. Programmers can modify existing classes or write their own new classes or they can write a whole new package. The JDBC API, the java.sql package, is one example of a foundation upon which extensions are being built. Other extensions are being added or worked on in area such as multimedia, Internet Commerce, conferencing, and telephony.

In addition to extensions there are also main tools being developed to make existing capabilities easier to use. For example, there is already a tool that greatly Simplifies creating and laying out Graphical User Interfaces such as menus, Dialog boxes and buttons.

#### SECURITY:

It is important that a programmer not be able to write subversive code for Applications or applets. This is especially true with the Internet being used more and more extensively for services such as electronic commerce and electronic distribution of software and multimedia content.

The Java platform builds in security in four ways.

* The way memory is Allocated and laid out: In Java an object’s location in memory is not determined until The runtime, as opposed to C and C++, where the compiler makes memory layout Decisions. As the result, a programmer cannot look at a class definition and figure out how it might be laid out in memory. Also since, Java has no pointers, a programmer cannot forge pointers to memory.
* The way incoming code is checked: The Java virtual machine doesn’t trust any incoming code and subjects it to what is called byte code verification. The byte code Verifier, part of the virtual machine, checks that the format of incoming code is correct

Incoming code doesn’t forge pointers, it doesn’t violate access restrictions, it accesses objects what they are.

* The way classes are loaded: The Java byte code loader, another part of the virtual machine, whether classes loaded during program execution are local or from across a network. Imported classes cannot be substituted for built in classes, and built in classes cannot accidentally reference classes brought in over a network.
* The way access is restricted for untested code: The Java security manager allows user to restrict untested Java applets so that they cannot access the local network, files and other resources.

#### JAVA PERFORMS WELL:

Java performance is better than one might expect. Java has many advantages, such as having built in security and being interpreted as well as compiled, do have a cost attached to them. However, various optimizations have been built in, and the byte code Interpreter can run very fast the cost it doesn’t have to do any checking. As a result, Java has done quite respectably in performance tests. Its performance numbers for interpreting byte codes are usually more than adequate to run interactive graphical end user applications.

For situations that require unusually high performance, byte codes can be translated on the fly, generating the final machine code for the particular CPU on which the application is running at run time. High level interpreted scripting language generally offer great portability and fast prototyping but poor performance. Low level compiled language like C and C++ offer great performance but require large amounts of time for writing and debugging code because of problems with areas such as memory management, pointers and multiple inheritances. Java offers good performance with the advantages of high level languages but without the disadvantages of C and C++.

**ROBUST:**

The multi plat formed environment of the WEB places extraordinary demands on a program, because it must execute reliably in a variety of systems. Thus the ability to create robust programs was given a high priority in the design of Java. To gain reliability, Java restricts you in a few key areas to force you to find your mistakes early in program developments. At the same time, Java frees you from having to worry about many of the most common cause of programming errors. Because Java is strictly typed language, it checks your code at compile time. However, it also checks your code at run time. In fact, many hard to track down bugs that often turn up in hard to reproduce runtime situations are simply impossible to create in Java. Knowing that what you have written will behave in a predictable way under diverse conditions is a key feature of Java to understand how Java robust.

Consider two main reasons for program failure:

* Memory management mistakes and mishandled exceptional conditions (run time errors).
* Memory management can be difficult, tedious task in traditional programming environments.

For example in C/C++ the programmer must manually allocate and free all dynamic memory. This sometimes leads to problems. For example some programmers some times forget the free memory that has been previously allocated. Or worse, they may free some memory that another part of their code is still using. Java virtually eliminates these problems by managing memory allocations and reallocations. Java helps in this area by providing object oriented exception handling. In a well-written Java a program should manage program all run time errors.

### SCALABILITY

Java platform is designed to scale well, from portable consumer electronic devices to powerful desktop and server machines. The virtual machine takes a small foot print and Java byte code is optimized to be small and compact. As a result, Java accommodates the need for low storage and for low bandwidth transmission over the Internet. In addition the Java operating system offers a standalone Java platform that eliminates host operating system overhead while still supporting the full Java platform. API makes Java ideal for low cost network computers whose sole purpose is to access the Internet.

#### MULTITHREADED:

Multithreading is simply the ability of a program to do more than one thing at a time. For example an application could be faxing a document at the same time it is printing another document. Or a program could process new inventory figures while it maintains a feed for current prices. Multithreading is particularly important in multimedia. A multimedia program might often be running a movie, running a audio track and displaying text all at the same time.

#### IMPORTANT TO THE INTERNET:

The Internet helped catapult Java to the forefront of programming and Java in turn has a profound effect on the Internet. The reason is simple. Java expands the universe of objects that can move about freely in cyberspace. In a network, there are two broad categories of objects transmitted between the server, your personal computer, passive information and dynamic, active programs. For example, when you read your e-mail, you are viewing passive data. Even when you download a program, the program’s code is still only passive data until you execute it. However, there is a second type of object that can be transmitted to your computer, a dynamic, self executing program. Such a program would be an active agent on the client computer, yet it would be initiated by the server. As desirable as dynamic, networked programs are, they also present serious problems in the areas of security and portability. Prior to Java cyberspace was effectively closed to half the entities that now live there. Java addresses these concerns and doing so, has opened the door to an exiting a new form of program.

**JAVA DATA BASE CONNECTIVITY:**

**INTRODUCTION:**

Java Database Connectivity (JDBC) is a front-end tool for connecting to a server to ODBC in that respect, However JDBC can connect only Java clients and it uses ODBC for the connectivity. JDBC is essentially a low-level application programming interface. It is called a low-level API since any data manipulation, storage and retrieval has to be done by the program itself. Some tools which provide a higher-level abstraction or expected shortly.

The next question that needs to be answered is why we need JDBC, once we have ODBC on hand. We can use the same ODBC to connect the entire database and ODBC is a proven technology. Problem for doing this is ODBC gives a ‘C’ language API, which uses pointers extensively. Since Java does not have any pointers and is object-oriented sun Microsystems, inventor of Java developed to suit its needs.

**REQUIREMENTS TO USE JDBC:**

To use JDBC you need a basic knowledge of database and SQL. Apart from this you need the jdk1.1 (Java Development Kit 1.1) or a version of Java since jdk1.1 and above come bundled with JDBC software.

After that you need to have a back-end database engine for which a JDBC driver is available. When JDBC drivers are not available JDBC-ODBC bridge drivers are used to access the database through ODBC. Back-end is not need when JDBC driver is capable of storing and retrieving the data itself, or if JDBC-ODBC bridge and the ODBC driver can be store and retrieve the information.

###### **DATABASE MODELS:**

JDBC and accessing the database through applets, and JDBC API via an intermediate server resulted in a new type of database model which is different from the client-servers through which the request should go it is named as single tier, two tier and multi tier architecture.

##### **JDBC DRIVER TYPES:**

***The JDBC drivers that we are aware of at this time fit into one of four categories:***

* JDBC-ODBC Bridge plus ODBC driver: The Java Soft bridge product provides JDBC access via ODBC drivers. Note that ODBC binary code and in many cases database client code must be loaded on each client machine that uses this driver. As a result, this kind of driver is most appropriate on a corporate network where client installations are not a major problem, or for application server code written in Java in three-tier architecture.
* Native-API partly-Java driver: This kind of driver converts JDBC calls into calls on the client API for Oracle, Sybase, Informix, DB2, or other DBMS. Note that, like the bridge driver, this style of driver requires that some binary code be loaded on each client machine.
* JDBC-Net all-Java driver: This driver translates JDBC calls into a DBMS-independent net protocol that is then translated to a DBMS protocol by server. This net server middle ware is able to connect its all-Java clients to many different databases. The specific protocol used depends on the vendor. In general, this is the most flexible JDBC alternative. It is likely that all vendors of this solution will provide products suitable for Internet use. In order for these products to also support Internet access, they must handle the additional requirements for security, access through firewalls, etc., that the Web imposes. Several vendors are adding JDBC drivers to their existing database middle ware products.
* Native-protocol all-Java driver: This kind of driver converts JDBC calls into the network protocol used by DBMS directly. This allows a direct call from the client machine to the DBMS server and is a practical solution for Internet access. Since many of these protocols are proprietary, the database vendors themselves will be the primary source. Several database vendors have these in progress.

Eventually, we expect the last two drivers will be preferred way to access database from JDBC. And the first two driver categories are interim solutions where direct all-Java drivers are not yet available. The last driver is in some sense the ideal one. However, there are many cases where JDBC-Net all-Java driver may be preferable. For example, where a thin DBMS- independent client is desired, or if a DBMS-independent protocol is standardized and implemented directly by many DBMS vendors.

# HTML

**INTRODUCTION:**

The Hyper Text Markup Language (HTML) is a simple markup language used to create hypertext documents that are portable from one platform to another. HTML documents are SGML documents with generic semantic that are appropriate for representing information from a wide range of applications. This specification defines HTML version 3.2. HTML 3.2 aims to capture recommended practice as of early ‘96 and as such to be used as a replacement for HTML 2.0(RF1866).

A set of instructions embedded in a document is called Markup Language. These instructions describe what the document text means and how it should look like in a display. Hyper Text Markup Language (HTML) is the language used to encode World Wide Web documents. It is a document layout and hyperlink specification language that defines the syntax and placement of special embedded directions that are not displayed by a web browser, but tells it how to display the contents of the documents including text, images and other supported media.

## USE OF HTML:

Web site is a collection of pages, publications, and documents that reside on web sever. While these page publications, and a document as a formatted in any single format. You should use HTML for home page and all primary pages and the site. This will enable the millions of web users it easily access and to take advantage of your website. HTML is considered first for formatting any new material you plan to publish on the web. HTML documents are platform independent, meaning that they don’t confirm to any standard. If they are created properly you can move home to any server platform or you can access them with any complaint www browser.

## BLOCK OF HTML:

### HTML elements perform a defined task. HTML uses two types of elements

* Empty tags(open tags)
* Container tags

These tags differ because of what they represent. Empty tags represent formatting constructs such as line breaks and Horizontal rules. Container tags define a section of text and specify the formatting the container dot all of the selected text. A container tag has both a beginning and an ending.

**HTML LAYOUT:**

An HTML document consists of text, which comprises the content of the document and tags which, defines the structure and appearance of the document. The structure of an HTML document is simple.

***<HTML>***

***<HEAD>***

***<TITLE> the title of the HTML document </TITLE>***

***</HEAD>***

***<BODY>***

This is where the actual HTML documents

Text lies which is displayed in the browser

***</BODY>***

***</HTML>***

5.3.4.1 PROGRAM DESCRIPTION:

The first line i.e., <HTML> tag, HTML tag is beginning tag and second line is starting tag for head section is <HEAD> The third line i.e., <TITLE> form example program </TITLE> is the title of the program. It defines a text string that is interpreted as the HTML title of the document. The tag </HEAD> will end the HEAD section of the program. Next tag is <BODY> the beginning of the body section where HTML document text lies, which is displayed in the browser. Next tags </BODY> </HTML> are the ending tags for the body section and html program respectively.

##### Each document has a head and body delimited by the <HEAD> and <BODY> tag. The head is where you give your HTML document a title and where you indicate other parameters the browser may use when displaying the document. The body is where you put the actual contents of the HTML documents. This includes the text for displaying the text. Tag also references special and hot spots that link your document to other documents.

**BACK END**

**MICROSOFT ACCESS**

**INTRODUCTION:**

Access is a database management system. Databases are an important part of everyday use. We use them anywhere we use computers, at home, and in the fields of medicine, education, law, business, engineering and government.

A database is a collection of data and facts that are related to a particular object or purpose, such as tracking customer orders or maintaining a music collection. If your database is not stored on a computer, or only parts of it are on the computer, you may be tracking information from a variety of sources that you have to coordinate and organize yourself.

In Access, the heart of the database is the Table. One or more tables comprise a database. In addition to tables, a database can contain other objects such as queries, reports, forms and other components. These objects allow you to view and manipulate the data contained in the database in variety of ways.

|  |  |
| --- | --- |
| Database Objects | Description |
| Table | The primary units of a database that stores field names, field description, field controls and field data. Tables display multiple records in a row/column format similar to a spreadsheet layout. |
| Query | A structured guideline used to search database tables and retrieve records that meet specific conditions. |
| Form | An aesthetically pleasing layout of table data designed to display one record on-screen at a time. |
| Report | An organized format of database data designed to generate printouts that provide meaningful information. |
| Pages | A data access page is a special type of Web page designed for viewing and working with data from an Internet or Intranet data that is stored in a Microsoft Access database or a Microsoft SQL Server database. The data access page may also include data from other sources, such as Microsoft Excel. |
| Macros | A mini program that stores a set of instructions designed to perform a particular task. |
| Module | A collection of Visual Basic programming procedures stored together to customize the Access environment. |

# DESIGNING A DATABASE:

Good design is a key factor of successful database building. A well designed database can lead to more efficient use of queries, forms, and reports, and can increase the reliability of the extracted information. In addition, an effectively designed database will be easier to expand as you organization’s information requirements grow and change.Although there are guidelines to assist you in designing your database, there is no such thing as a perfect database design. Designing information systems always involves a degree of compromise. The purpose of this appendix is to provide a general overview of good database design principles. These guidelines should not be viewed as hard and fast rules, but rather suggestions for making your database design decision.

**DATABASE DESIGN PROCESS:**

Although there is no one correct way to create a database, there are some general steps that some professional systems developers follow. These steps generally reflect what is known as the Systems Development Life Cycle (SDLC). The SDLC breaks the task of creating a system into stages.

A variety of techniques can be employed to answer these questions. One of the best ways is to involve the future end-users of the database. A database, however, may have hundreds or even thousands of users. However, if the database is very large, you might have to consult representatives of various user groups, such as a database design committee.

# WORKING WITH TABLES IN DATASHEET VIEW:

Tables are used to store data in a database. Tables can be created in two views, Table Design view or Datasheet view. In addition, you can use the Table Wizard or import a table from an external file. Records are automatically saved as you move from record to record but changes to the layout of a table (or a form) must be saved.

**TABLE DESIGN VIEW:**

In table design View, you must specify the fields, specify the data type for each field, assign the size for text and number fields, assign the primary key (see glossary for definition) and save the table yourself.

**DATASHEET VIEW:**

In data sheet view data types are very important to creating a database. There are 10 different types to choose from. The data type determines what kind of information may be entered in a field. You can always change the data type of a field after you have entered data into it. However, it may be a difficult process if the data field conflicts with the current data type (for example, changing a text field to a number field). The table below shows the specifics on each data type.

|  |  |
| --- | --- |
| Setting | Type of Data |
| Text(default setting for all field) | Text or combination of text and numbers, as well as numbers that don’t require calculations, such as phone numbers. Up to 255 characters or the length set by the FieldSize property, whichever is less. |
| Memo | Lengthy text or combinations of text and numbers. Up to 65,536 characters. |
| Number | Numeric data used in mathematical calculations. |
| Date / Time | Date and time values for years 100 through 9999. |
| Currency | Currency values and numeric data used in mathematical calculations involving data with one to four decimal places. Accurate to 15 digits on the left side of the decimal separator and to 4 digits on the right side. |
| AutoNumber | A unique sequential (incremented by 1) number or random number assigned by Microsoft Access whenever a new record is added to a table. AutoNumber fields cannot be updated. |
| Yes / No | Yes and No values and fields that contain only one of two values (Yes/No, True/False, or On/Off). |
| OLE Object | An object (such as a Microsoft Excel spreadsheet, a Microsoft Word document, graphics, sound, or other binary data) linked to or embedded in a Microsoft Access table. |
| Hyperlink | Text or combination of text and numbers stored as text and used as hyperlink address. |

# 5.4.3 WIZARDS:

Access provides several wizards that you can use to create databases for business or personal use. This is a wonderful time saver. There are 25 sample tables for business and personal use. Once you select a template, you can modify the fields to better suit your needs. In addition, you can rename fields to more accurately define them. Access can also provide samples in the database. Samples may help you learn to use the database faster.

**JAVA SERVER PAGES (JSP)**

**INTRODUCTION:**

Java Server Pages (JSP's) permit server side Java logic to reside within the requested document. Upon request of a JSP document the server activates the specified JSP. The JSP then becomes responsible for providing an HTML response.

The server side logic within a JSP is written in Java. The Java code segments, referred to as scriptlets, are generally responsible for providing dynamic HTML content to the JSP's response HTML. The JSP itself is compiled by the server, and is executed as an object that extends the Java Servlets API. As such, the HTTP Servlet request and response objects are available by the script lets defined within the JSP.

This document reviews client-server design considerations in respect to the use of JSP’s. Implementation options, particularly the use of JSP language extensions and use of Enterprise Java Beans (EJB's) will also be discussed. Focus will be placed on the presentation layer and how the JSP is used to provide a user interface and communicate business logic requests to the supporting system.

If we consider a 3-tier architectural WEB application, the browser becomes the client side application. The user communicates requests to the WEB/app server via the browser. The presentation layer receives the client requests and prepares the response and server side business functionality is executed. In the context of this example, the JSP engine represents the presentation layer. It is responsible for processing requests and responses. Additional messages may be passed between this layer and that which handles business processes represented below as EJB’s.

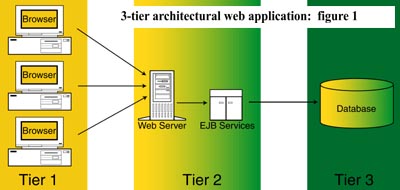


FIGURE1

**THE TECHNOLOGY:**

JSP technology uses XML - like tags and scriptlets. They are used to encapsulate presentation logic within the JSP. They can also initiate messages to distributed or server-side applications. The logical separation of presentation and business logic lies in the implementation of the JSP.

Enterprise Java Beans provide a distinct relationship between the implementation of business logic and the remote interfaces provided to the EJB client. The use of an EJB typically follows the pattern:

* The client application identifies itself to the server.
* The client application uses the Java Naming Directory service to locate the desired EJB.
* The client application retrieves a handle to the EJB Home and subsequently Remote interfaces.

The remote interface contains methods that the client is permitted to use. They represent a summary of the business logic that is implemented by the bean. The implementation logic is defined within the primary bean class. All IPC, database and resource details are restricted to the bean class.

In constructing a JSP document, the creation of the HTML base is a prudent step. It becomes the visual template that JSP scriptlets are merged into. The post execution HTML produced from the completed JSP should be that of the original HTML document. With the exception of comment, dynamically generated HTML sections and JSP content substitutions. The scripting logic, except for where desired, is completely non visual in regard to the response HTML text.

The construction of the HTML layout conceivably begins with a Web developer. The creation of the JSP pages would be similar if not identical to the methods used to construct industry HTML pages. The next step would be the addition of JSP specific logic to identify the sections of the HTML that might be generated dynamically. This conversion step from pure HTML to JSP is where server side logic is added to the page.

A completed JSP logically embodies presentation layer services and business functionality. Physically they are blended within the JSP in an as needed swapping of HTML and JSP code. Continued maintenance of the application and changes in the business logic need not affect the presentation layout. Likewise, changes in the presentation layout need not affect the scriptlet logic, it will however require that the WEB developer, not necessarily a JAVA programmer, show care in the handling of this file which is no longer pure HTML should any HTML maintenance become necessary.

**THE ALTERNATIVE:**

A design consideration intended to reduce the complexity of maintaining the HTML aspect of a JSP is to minimize the use of scriptlets in constructing a JSP. Custom tags, introduced in JSP 1.1, can equally produce the functionality provided by JSP scriptlets.

Custom tags are application defined language extensions to Java Server Pages. Custom tags can be used within a JSP in the following ways:

* To produce html output.
* To produce JSP output (JSP expressions, directives, ...).
* To create objects.
* To define objects that can be seen as scripting variables within the parent JSP.
* To iterate over a body of JSP/HTML text in a finite manner.
* To determine if section of the calling JSP should be processed or skipped.

The goal of using custom tags to minimize the presence of scriptlets is to produce a more HTML – like JSP. The advantages of this goal are self-evident if we consider projects that expect frequent HTML modifications. Assuming the business logic, pre-presented by the JSP tags, is stable it can be identically merged into various forms of the HTML layout, without explicitly inserting duplicate sections of scriptlet logic (Java code).

Tag handlers implement JSP custom tags. One or more tag handlers can be listed in the Tag Library Descriptor files. References to these files are included in the JSP that intends to use a given tag handler. The tag handler itself is implemented as a Java object that extends the JSP body. Upon execution it has access capabilities to the JSP's Http servlet objects, page attribute and session attribute objects. It can, conceivably, provide a full HTML response to the client in the way that servlets operate. A significant distinction from Java Server Pages is that tag handlers are not designed to be dynamically compiled by the server.

In respect to EJB's, a tag handler accesses an EJB in the same manner as the above scriptlet. It can additionally make available any object it creates, available to other tag handlers and JSP’s. This is accomplished by the use of storage methods that operate within the scope of the page and session. This includes the retention of EJB remote interface objects that can be created once and re-used by subsequent JSP’s via scriptlets or tags.

The JSP engine and Java Server Pages logically produce presentation layer services. They also provide the interface to business services (i.e. EJB’s). The physical separation of the logic associated with these middle tier components is evident in the above example. The same EJB logic in the previous example is represented here by the tag references.

Figure 2 gives a graphical representation of the physical control flow without the use of custom tags. The client initiates execution with a JSP request. The request via URL is directed to the WEB server that is responsible for servicing such requests. The JSP request triggers the JSP engine to locate and execute the corresponding JSP as a servlet object. The execution of the business logic is represented by the use of Enterprise Java Beans.

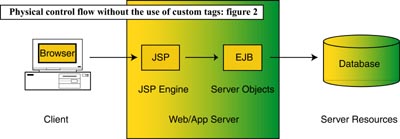


Figure 2

Logically identical, figure 3 illustrates the use of tag handlers by the JSP. This is the hidden logic implied in HTML example 2.

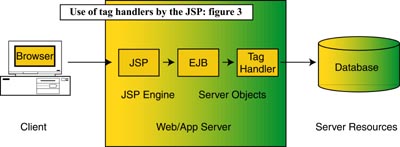


Figure 3

The JSP engine, in both figures, treats the compiled JSP object as a servlet object. Figure 3’s tag handler object extends the JSP page body. This relationship grants tag handler access to various Servlets attributes. These attributes therefore permit the tag handler to conceivably inspect parameters passed by the client.

**CONCLUSION:**

As with other tools of the trade, innovations and nuances to existing tools do not invalidate existing design methodologies. They do however provide new versatility and the expansion of possibilities with regard to application design.

Custom tag extensions, in contrast to standard tags, provide the application builder the ability to define custom tags to satisfy some functionality not provided by the standard API. To benefit by using tag extensions to reduce the amount of Java functionality that the JSP API provides, might seem oxymoronic, and it is. With the exception of dynamically compiled JSP’s, the functionality provided by the two given examples are identical, which suggests that the payoff for implementing this server side alternative is purely cosmetic, and it is.

While a server side application designer does not typically consider the cosmetic aspect of implementing source code, JSP source code might prove to be the exception. It does after all suggest the strong possibility that a Web/HTML developer perform the continued maintenance of the HTML portion of the JSP. This is a role, of course, traditionally allied with client side responsibilities.

The nuances introduced by JSP custom tags present nuances in the maintenance of JSP’s. The arguments presented here presume that the HTML produced by the JSP’s in discussion are non-trivial HTML documents, although non-complex HTML documents may benefit from similar design considerations

**TESTING AND IMPLEMENTATION**

**6.1 Test Case Description**

**6.1.1 Definition**

Testing is the process of detecting errors. Testing performs a very critical role for quality assurance and for ensuring the reliability of software . The results of testing are used later on during maintenance also.

6.1.2 Psychology of Testing

The aim of testing is often to demonstrate that program works by showing that it has no errors. The basic purpose of testing phase is to detect the errors that may be present in the program. Hence one should not start testing with the intent of showing that a program works, but the intent should be to show that a program doesn’t work. Testing is the process of executing a program with the intent of finding errors.

6.1.3 Testing Objectives

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say,

• Testing is a process of executing a program with the intent of finding an error.

• A successful test is one that uncovers an as yet undiscovered error.

• A good test case is one that has a high probability of finding error, if it exists.

• The tests are inadequate to detect possibly present errors.

• The software more or less confirms to the quality and reliable standards.

**6.2 Types of Testing**

**6.2.1 Unit Testing**

Unit testing focuses verification effort on the smallest unit of software i.e. the module. Using the detailed design and the process specifications testing is done to uncover errors within the boundary of the module. All modules must be successful in the unit test before the start of the integration testing begins.

In this project each service can be thought of a module. There are three basic modules. Giving different sets of inputs has tested each module. When developing a module as well as finishing the development so that each module works without any error. The inputs are validated when accepting from the user.

In this application developer tests the programs up as system. Software units in a system are the modules and routines that are assembled and integrated to form a specific function. Unit testing is first done on modules, independent of one another to locate errors. This enables to detect errors. Through this, errors resulting from interaction between modules initially avoided.

**6.2.2 Link Testing**

Link testing does not test software but rather the integration of each module in system. The primary concern is the compatibility of each module. The Programmer tests where modules are designed with different parameters, length, type etc.

**6.2.3 Integration Testing**

After the unit testing we have to perform integration testing. The goal here is to see if modules can be integrated properly, the emphasis being on testing interfaces between modules. This testing activity can be considered as testing the design and hence the emphasis on testing module interactions. In this project integrating all the modules forms the main system. When integrating all the modules I have checked whether the integration effects working of any of the services by giving different combinations of inputs with which the two services run perfectly before integration.

**6.2.4 Code Testing**

This strategy examines the logic of the program. To follow this method we developed some test data that resulted in executing every instruction in the program and module i.e. every path is tested. Systems are not designed as entire nor are they tested as single systems. To ensure that the coding is perfect two types of testing is performed or for that matter is performed or that matter is performed or for that matter is performed on all systems.

**6.2.5 Validation Testing**

Validation can be defined in many ways, but a simple definitions is that validation succeeds when software functions in a manner that can be reasonable expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirements.

**6.2.6 Black Box Testing**

Black Box Testing attempts to find errors in following areas or categories, incorrect or missing functions, interface error, errors in data structures, performance error and initialization and termination error. Here all the input data must match the data type to become a valid entry.

**6.2.7 White Box Testing**

White box testing is a testing case design method that uses the control structure of the procedure design to derive test cases. All independents path in a module are exercised at least once, all logical decisions are exercised at once, execute all loops at boundaries and within their operational bounds exercise internal data structure to ensure their validity. Here the customer is given three chances to enter a valid choice out of the given menu. After which the control exits the current me

**6.6 System Documentation:**

All design and test documentation should be finalized and entered in the library for future reference. The library is the central location for maintenance of the new system. The format, organization, and language of each documentation should be in line with system standards.

**TEST CASES**

**Module: Login**

**Filename: form1.vb**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Input** | **Actual Output** | **Obtained Output** | **Desc** |
| **Valid Login** | **User Id, Password** | **Login Success** | **Login Success** | **Test Passed!**  **Control Transferred to Menu** |
| **Invalid Login** | **User Id, Password** | **Login Failed** | **Login Failed** | **Test Passed!**  **Try Again** |
| **Invalid Login** | **Null,Null** | **Login Failed** | **Login Failed** | **Test Passed!**  **Try Again** |

**Module: Citizen Regn**

**Filename: Czregn.vb**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case | Input | Actual Output | Obtained Output | Desc |
| Register Citizen | Citizen Info, Card ID | Success | Success | Test Passed.  Citizen Regd and associated with Aadhaar card. |
| Register Citizen | Citizen Info, Card ID | Failed | Failed | Test Passed.  Invalid or incomplete data. Try again. |

**Module : Passport**

**Filename : passport.vb**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case | Input | Actual Output | Obtained Output | Desc |
| Passport Status | Citizen ID, pincode | Success | Success | Test Passed.  Issue passport, associate expiry info. |
| Passport Status | Citizen ID, pincode | Failed | Failed | Test Passed.  Invalid data or pincode incorrect or police verification failed. |

**Module : License**

**Filename : license.vb**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case | Input | Actual Output | Obtained Output | Desc |
| License Status | Citizen ID, Pincode, Test Info, vtype | Success | Success | Test Passed.  License issued & saved . |
| License Status | Citizen ID, Pincode, Test Info, vtype | Failed | Failed | Test Passed.  Invalid data or test failed. Try Again |

**Future Scope:-**

1. Usage of biometric devices to identify citizen.
2. Usage of the card to perform financial transactions.
3. PAN, Voter ID, Ration Card etc can also be processed by using this card.

**Limitations**

1. Dependency on the hardware kit to maintain an alternative PCB at all times.
2. No facility to receive mobile alerts.
3. More number of Aadhaar seva centers required.

**CONCLUSION**

**CONCLUSION**

The application can now identify each individual uniquely. Every citizen is identified for all the Govt transactions with the help of his Aadhaar card. The application integrates various Govt departments into a single point of Contact. This helps in avoiding unnecessary delays or find where the delay is happening when applications are processed. The application can be extended to all the Govt departments with modification. New modules can be added without affecting the existing modules.

**BIBLIOGRAPHY**

**BIBLIOGRAPHY:**

* [**www.Java.sun.com**](http://www.Java.sun.com)
* [**www.students3k.com**](http://www.students3k.com)
* **The Unified Modeling Language User Guide**
* **Java Server Page-“O reliey”**
* **Java-Complete Reference**