

APPENDIX A

1. Introduction

1.1. BACKGROUND

Personal Information Manager is a java application for mobile devices. It uses database to store contact information, day notes and to-do lists, and provides easier way to manage these information.

1.2. OVERALL DESCRIPTION

At current scenario, mobile devices implement contact information management and day-note management separately. Current mobile devices allows the user to store contact name and 3-4 contact numbers only on average mobile phones and additional information in case of advanced mobile devices.

In case of day-notes, current system is not being able to avoid the collision among the meeting schedules. Birthday reminders are also needed to be entered separately in current system. Current system search function allows searching by name only but not by other fields of information such as phone number, contact group etc.

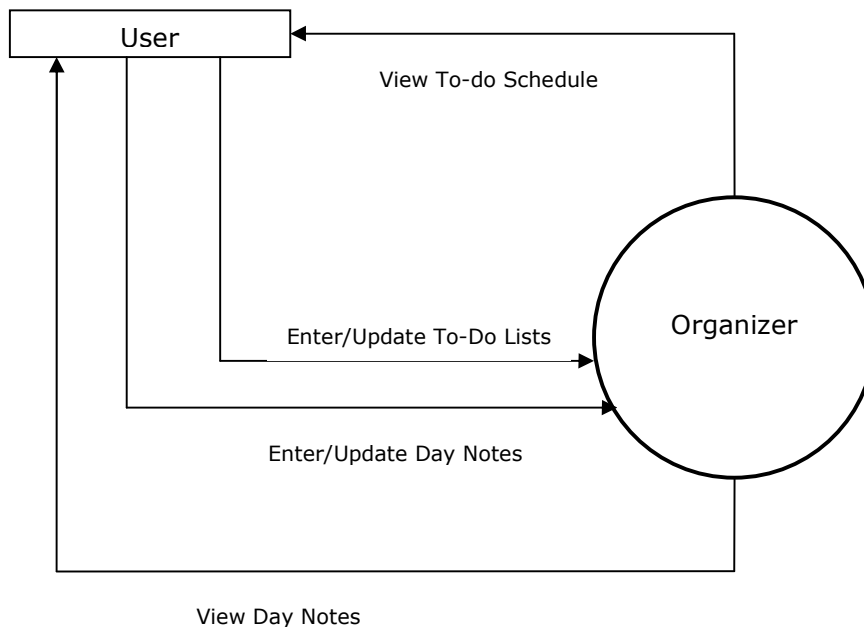


FIG1.CONTEXT DIAGRAM OF CURRENT SYSTEM

The new system to be developed must integrate the organizer and contact information. The other requirement for the new system would be to avoid invalid meeting and problematic meeting schedules. Allowing different search options would be other necessity of the new system to be developed. Integrating To-do list and day notes would also be other concern for the new system.

2. GLOSSARY

Figures:

- FIG1: Context Diagram of Current System
- FIG2: Data Flow Diagram for New System
- FIG3: Use Case Diagram
- FIG4: List Interface for User
- FIG5: Form Interface for User
- FIG6: System Architecture of New System

3. INFORMATION FLOW

Personal Information Manager has the three resources of the information to be maintained and processed, and they include contact information, day notes and to-do lists. This information would be stored in relational database and are information from current system. This new system is responsible for managing the available information and newer information to be added efficiently.

3.1. DATA FLOW DIAGRAM

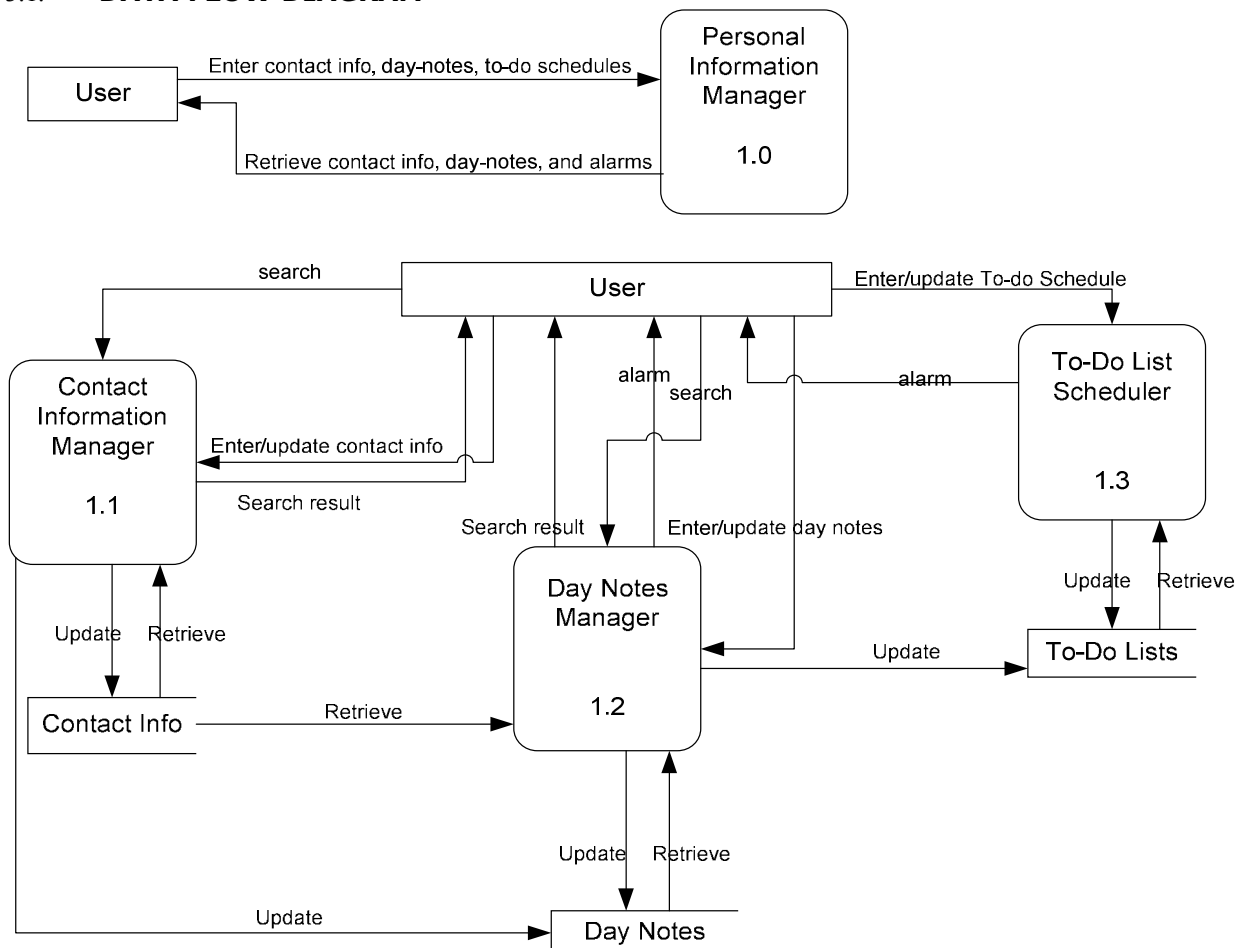


FIG2. Data Flow Diagram of Current System

Software Requirement Analysis

3.2. **CONTROL FLOW AND DESCRIPTION**

In the new system, for the better information organization proper flow of control is required among all three components that are contact manager, day-note manager and to-do list scheduler.

- Whenever day notes are entered, then the to-do list of that day in particular should also be updated automatically.
- As user chooses to keep a birthday reminder whenever entering the contact information then day notes should also be automatically updated.
- To enter meeting schedules on same day then day note manager should check collision first then only update the day notes

4. **FUNCTIONAL DESCRIPTION**

4.1. **USER REQUIREMENTS DEFINITION**

- Enter birthday notes from contact information
- Avoid meeting schedule collision
- Integrate day-notes and to-do lists
- Make contact information available for day notes and to-do lists
- Provide multiple search options for contact information and day notes

4.2. **FUNCTIONAL PARTITIONING**

The following functions are required to meet above requirements:

- addNewContact()
- updateContact()
- addBirthdayNote()
- addNewNote()
- updateNote()
- checkCollision()
- addToDoList()
- updateToDoList()
- insertToDoList()
- insertFromContact()
- searchContact()
- searchNote()
- searchByName()
- searchByNumber()
- searchByGroup()
- searchByDate()
- searchByType()
- viewToDoList()

Software Requirement Analysis

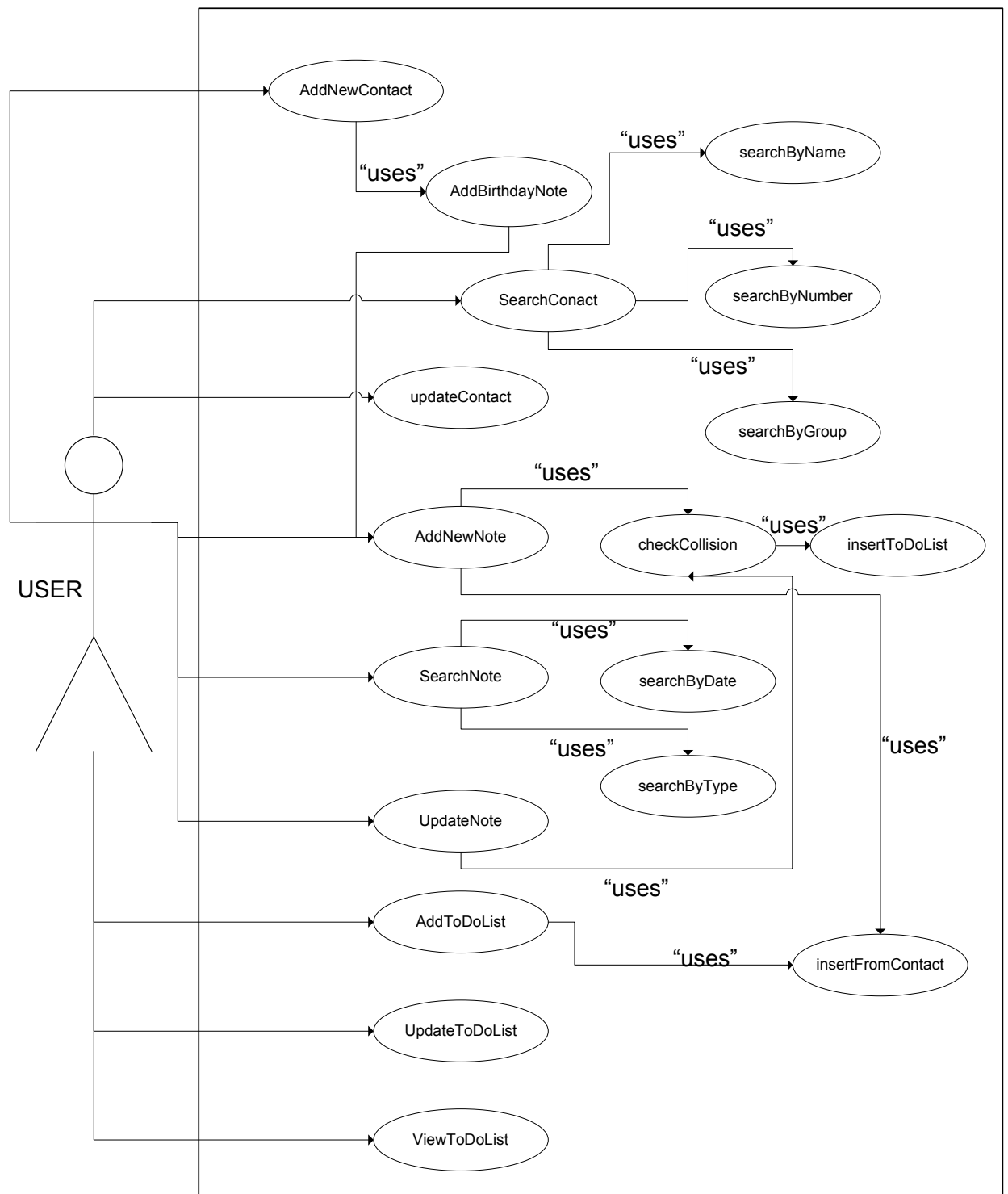


FIG3. USE CASE DIAGRAM

4.3. **FUNCTIONAL DESCRIPTION**

Following are the form-based description of the required functions:

- **addBirthdayNote()**

Function: Add Birthday Note
Description: Adds a birthday note of a contact to a specific date as a Day Note
Inputs: Birth date, contact name, time of alarm
Source: User Input
Outputs: A Birthday note
Destination: Day Note database
Requires: Day Notes
Pre-condition: None
Post-condition: None

- **addNewContact()**

Function: Add New Contact
Description: Allows user to enter new contact information
Inputs: Contact name, Contact numbers, Birth date, Contact group
Source: User Input
Outputs: None
Destination: Contact Database
Requires: None
Pre-condition: All fields except birthday and contact group should be filled
Post-condition: Contact name should not exist previously

- **updateContact()**

Function: Update Contact
Description: Allows user to update existing contact information
Inputs: Contact name, Contact numbers, Birth date, Contact group
Source: User Input
Outputs: None
Destination: Contact Database
Requires: SearchContact Function
Pre-condition: Contact information should exist previously
Post-condition: None

- **addNewNote()**

Function: Add Day Note
Description: Allows user to enter new day notes i.e. reminders, meetings, memo etc.
Inputs: Date of note, Note type, Note information
Source: User Input
Outputs: None
Destination: Day Notes Database
Requires: None
Pre-condition: All fields should be filled
Post-condition: None

Software Requirement Analysis

- updateNote()

Function: Update Day Note
Description: Allows user to update existing day notes
Inputs: Note information, Note Date pre-pone or postpone
Source: User Input
Outputs: None
Destination: Day Notes Database
Requires: searchNote function
Pre-condition: None
Post-condition: Day note should exist previously

- checkCollision()

Function: Check Schedule Collision
Description: Checks whether new meeting schedule collides with existing ones
Inputs: date of meeting, time of meeting
Source: AddNewNote function and UpdateNote function
Outputs: Favorable or acceptable or unacceptable
Destination: User
Requires: None
Pre-condition: None
Post-condition: None

- addToDoList()

Function: Add To Do List
Description: Allows user to enter a new to-do schedule
Inputs: date, task information, priority
Source: User
Outputs: None
Destination: To-Do List
Requires: None
Pre-condition: None
Post-condition: None

- updateToDoList()

Function: Update To Do List
Description: Updates an existing record in To-Do List database
Inputs: Task information, date and priority
Source: User
Outputs: None
Destination: To Do List
Requires: None
Pre-condition: None
Post-condition: None

Software Requirement Analysis

- insertToDoList()

Function: Insert to To-Do List
Description: Add to To-Do List from a day notes entered
Inputs: Day Note
Source: Day Note Database
Outputs: None
Destination: To Do List
Requires: None
Pre-condition: None
Post-condition: None

- insertFromContact()

Function: Insert from Contact
Description: Use contact information to make day notes and To-Do Lists
Inputs: Contact Name or Contact Numbers
Source: Contact Database
Outputs: None
Destination: To Do List, Day Notes
Requires: SearchContact
Pre-condition: None
Post-condition: None

- searchContact()

Function: Search Contact Information
Description: Search for contact information as per user's search option
Inputs: None
Source: None
Outputs: Contact Search Menu
Destination: User
Requires: searchByName, searchByNumber and searchByGroup Functions
Pre-condition: None
Post-condition: None

- searchNote()

Function: Search Day notes
Description: Search day notes either by date or by type of day note
Inputs: None
Source: None
Outputs: Note Search Menu
Destination: User
Requires: searchByDate and SearchByType Functions
Pre-condition: None
Post-condition: None

Software Requirement Analysis

- `searchByName()`

Function: Search By Name
Description: Search Contact Information by contact name
Inputs: Contact name
Source: User
Outputs: Search result
Destination: User
Requires: None
Pre-condition: None
Post-condition: None

- `searchByNumber()`

Function: Search By Number
Description: Search Contact Information by contact number
Inputs: Contact number
Source: User
Outputs: Search result
Destination: User
Requires: None
Pre-condition: None
Post-condition: None

- `searchByGroup()`

Function: Search By Group
Description: Search Contact Information by contact group
Inputs: Contact group
Source: User
Outputs: Search result
Destination: User
Requires: None
Pre-condition: None
Post-condition: None

- `searchByDate()`

Function: Search By Date
Description: Search Day Notes as per date
Inputs: Date
Source: User
Outputs: Search result
Destination: User
Requires: None
Pre-condition: None
Post-condition: None

Software Requirement Analysis

- `searchByType()`

Function: Search By Type
Description: Search Day Notes as per note type
Inputs: Note Type
Source: User
Outputs: Search result
Destination: User
Requires: None
Pre-condition: None
Post-condition: None

- `viewToDoList()`

Function: View To-Do Schedule
Description: Displays current day To-Do Schedule
Inputs: None
Source: None
Outputs: To-Do List of the day
Destination: User
Requires: None
Pre-condition: None
Post-condition: None

Among these required functions, current system provides the functions to enter contact information, day notes and to-do schedules. Also functions to update this information are also there in current system. Search functions to search contact information by name only is available. But remaining functions are the major requirements of the new system to be developed.

5. INTERFACES

5.1. HARDWARE INTERFACE

Personal Information Manager need not communicate with any other hardware so no hardware interface is required.

5.2. USER INTERFACE

New system would communicate with user through the graphical menus and forms which would look like following:



FIG4. LIST INTERFACE

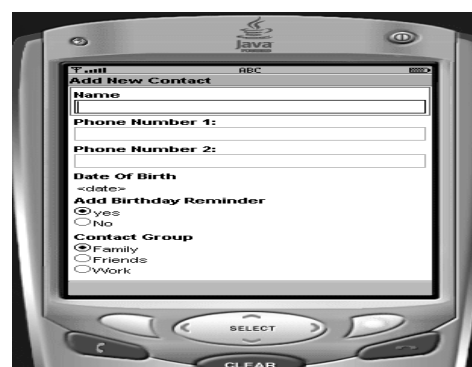


FIG5. FORM INTERFACE

5.3. INTERFACE WITH OTHER SYSTEM

New system would look much more attractive if it succeeds to make an API call to the vibration mode in the mobile devices. But even if it doesn't succeed the system would function efficiently by the alarm signals only.

6. SYSTEM ARCHITECTURE

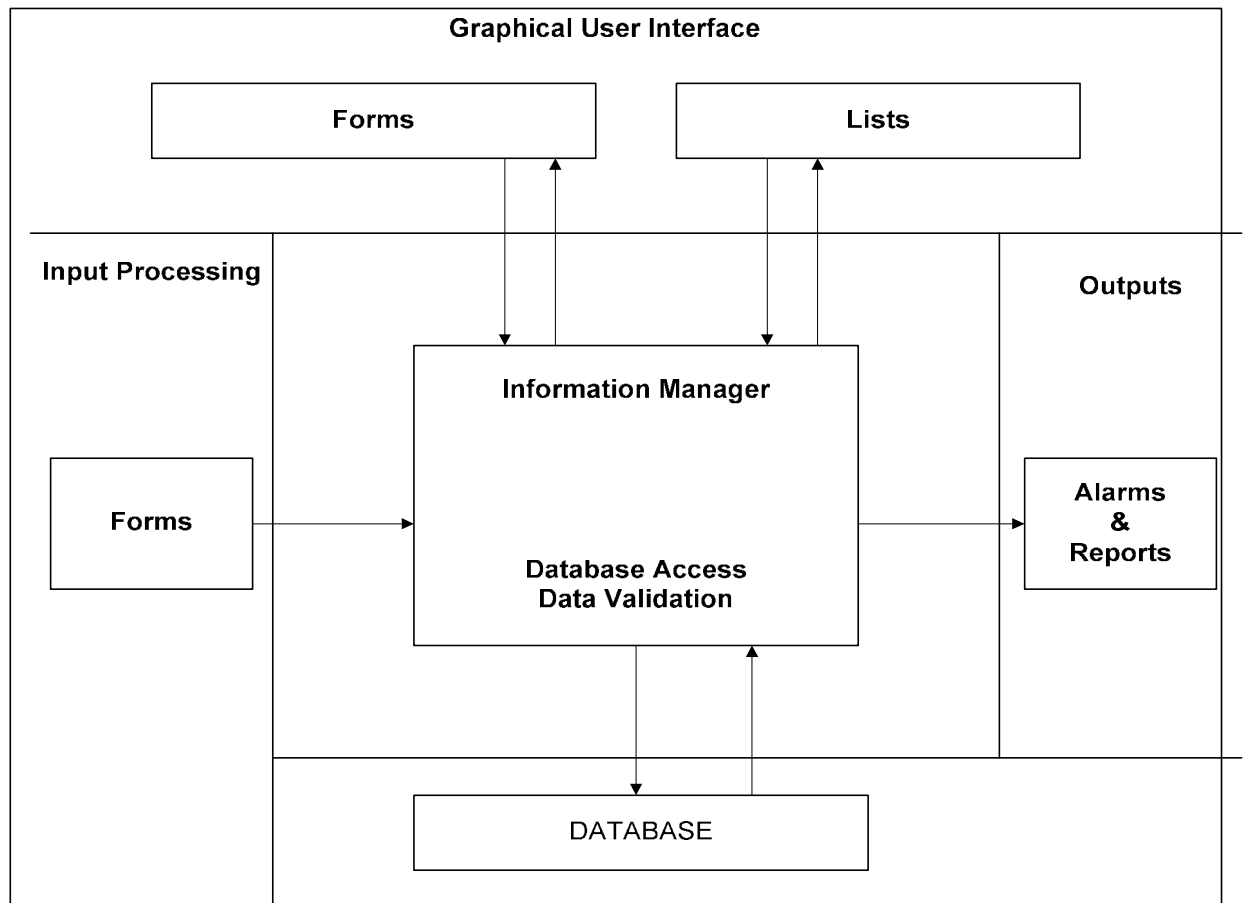


FIG6. SYSTEM ARCHITECTURE OF NEW SYSTEM

7. SYSTEM REQUIREMENTS SPECIFICATION

The new system provides:

- Facility to store contact information along with birthday
- Facility to add birthday reminder automatically
- Facility to search contact information by name, number and group
- Facility to enter different day notes without collision during meeting schedules
- Integration between Day notes and To-Do Schedule

8. NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements desired for this system are:

- Efficient rescheduling the day notes that includes pre-pone and postpone
- Vibration during alarm signals

9. SOFTWARE CONSTRAINTS

Since system is to be developed for mobile devices, the language to be used would be J2ME language and the backend database would be the implementation of Java DB. Tool to be used for this system "

10. BEHAVIOURAL DESCRIPTION

Any undesirable state is unexpected for the new system.

11. VALIDATION CRITERIA

11.1. PERFORMANCE BOUNDS

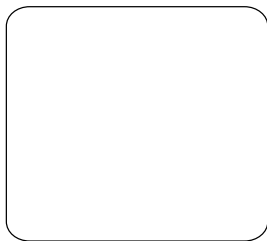
The new system should be around 2MB in size to be compatible for general mobile devices as well.

11.2. CLASS OF TESTS

Structural White Box testing including path testing should be accomplished successfully so as to be accepted by the customer.

12. APPENDICES

Data Flow Diagram Notation



Process



Interface



Data Store



Flow of Process