**DATA FLOW DIAGRAM (DFD)**



**TXTSCAN - OCR Based Android Application**

SUBMITTED TO: SUBMITTED BY:

Mrs. SUGANDHA SHARMA Harneet Singh

16bcs1624

PROJECT MENTOR:

Ms. Samiksha Dogra

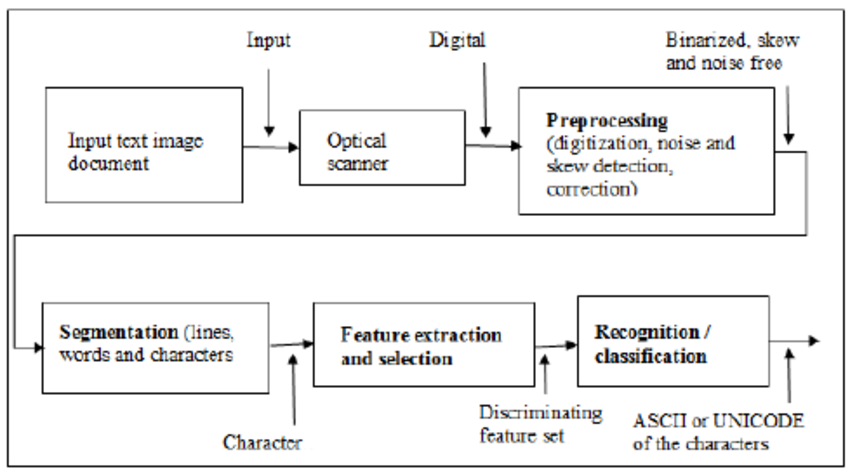
**TABLE OF CONTENTS FOR A DATA FLOW DIAGRAM**

1. Introduction
2. Data flow diagram
3. System Architecture
4. ER Diagram

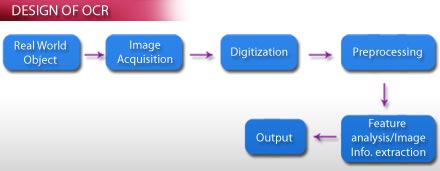
**Data Flow Diagram (DFD)**

**Introduction**

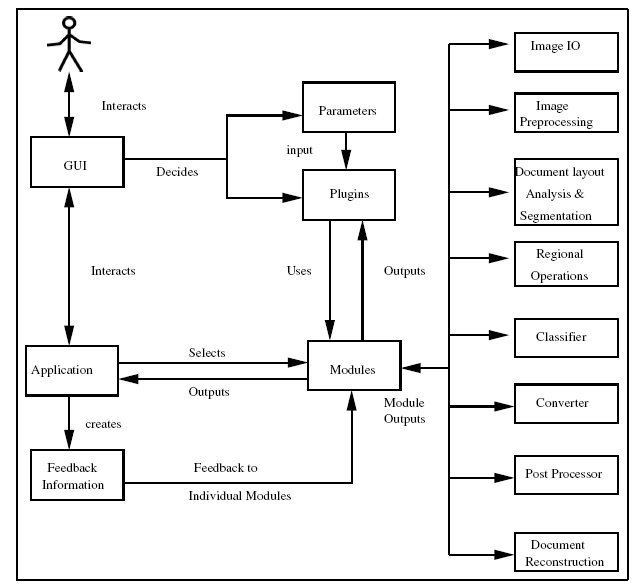
A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That’s why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

****

**DESIGN OF OCR**

****

**SYSTEM ARCHITECTURE**

****

In this a semi-automatic and adaptive system is implemented. Applicability of our design for the recognition of Indian Languages is demonstrated. Recognition errors are used to train the OCR again so that it adapts and learns for improving its accuracy. Limited human intervention is allowed for evaluating the output of the system and take corrective actions during the recognition process.

We design the general architecture of an interactivemultilingual OCR (IMOCR) system that is open for learning and adaptation. An overviewof the architecture of the system is shown in Figure. The IMOCR design is based on a multi-core approach. At the heart of the same is an application tier, which acts as the interface between the Graphical User Interface (GUI) and the OCR modules. This application layer identifies the user-made choices, initialises data and document structures and invokes relevant modules with suitable parameters. The GUI layer provides the user with the tools to configure the data-flow, select the plug-ins and supply initialization parameters. The system provides appropriate performance metrics in the form of graphs and tables for better visualization of the results of each step and module during the recognition process.

The last layer is the module/algorithm layer where the actual OCR operations are done. This layer is segmented based on clearly identified functionality. Each module implements a standard interface to be invoked via the application. Each module internally can decide on multiple algorithm implementations of the same functionality that may be interchanged at run-time. This helps in selection and use of an appropriate algorithm or a set of parameters for a book, collection or script. This layer is designed on the principle of plug-ins. The system allows transparent runtime addition and selection of modules (as shared objects/ dynamic libraries) thereby enabling the decoupling of the application and the plug-ins. Feature addition and deployment is as simple as copying the plug-in to the appropriate directory. The other advantages of this approach are lower size of application binary, lower runtime memory footprint and effective memory management (through dynamic loading and unloading, caching etc.).

**ENTITY RELATIONSHIP(ER Diagram)**

