**Team Project 1: Relational Database Management System Design Document**

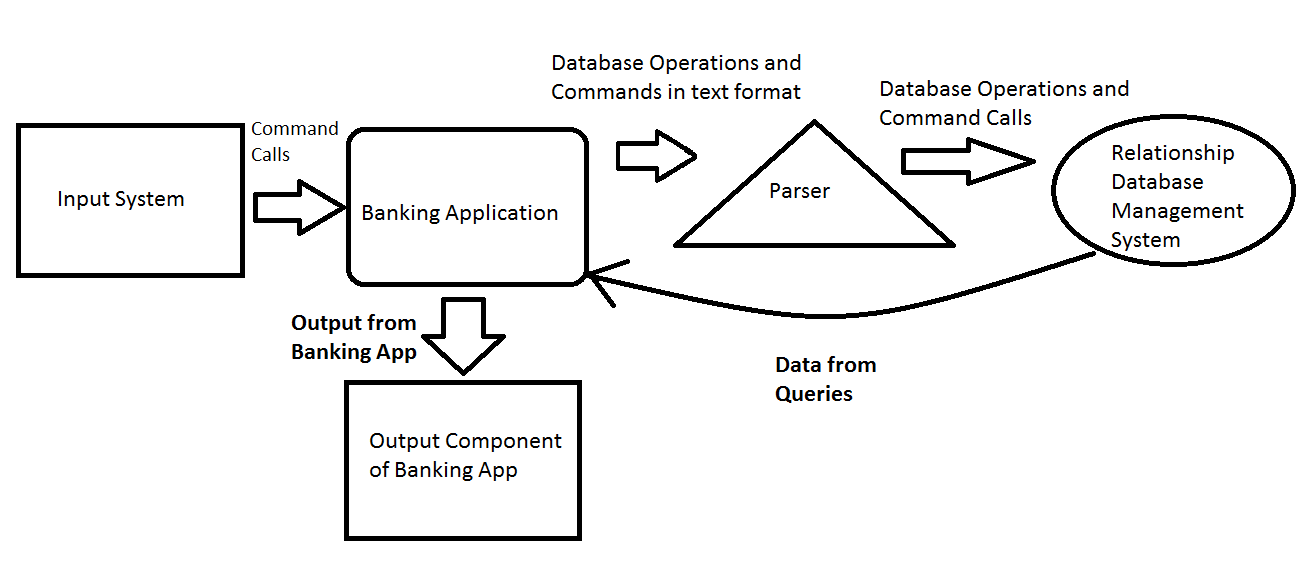
**Section 1 – Purpose:**

We are designing a Database Management System based on relational algebra which will power a simple application that will manage a simple bank database. The scope of this project is not to create an actual implementation of a Database Management System that any bank would actually use, so our application will only create databases with information about clients, their accounts, and deposits and money transactions.

The Database Management System engine has a grammar of its own, so it can, in principle, be used by an application written in any language that can interface with it. The mechanism for using the Database Management System engine is passing messages to an object that represents the engine, or possibly by running a text file through the engine (like a script through an interpreter).

The application is not really aimed at anyone, but it is a mere exercise for learning about databases. The problem “it solves” is keeping track of valuable information about a bank, its clients, and the money involved. The Database Management System engine is, also, an academic exercise, but it can be adapted for use with many application frontends. The academic purpose of this project is for us, the designers, to learn how to work in groups by becoming familiar with version control systems and other quirks of designing software in a group.

**Section 2 - High Level Entities:**



Input System: Accepts user input and makes the appropriate command calls to the Banking Application.

Parser: This component receives text queries from the application and converts them into relational database operations and commands that our database management system understands.

Relationship Database Management System: This system supports relational database operations and commands. This component and its API are used by the Banking Application to manage data. This component is also responsible for printing relations to plain text file and opening relations from plain text file.

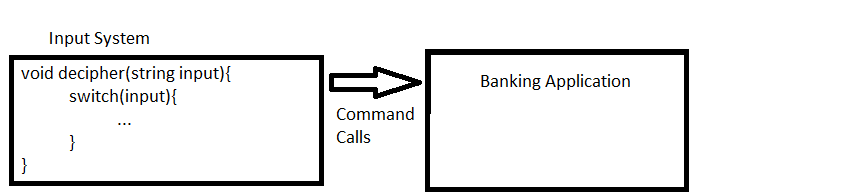
Output Component: This component will likely be integrated into the Banking Application and will be used to output application menus and the results of user input.

Banking Application: This is the primary component of the system; it is responsible for using the database management system to complete the input command calls. It is also responsible for providing data and context to the output system.

**Section 3 - Low Level Design:**

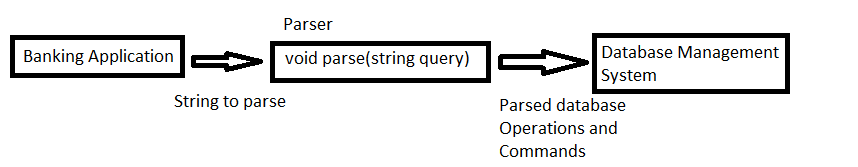
Input System:

Usage: This component accepts input from the standard in of console. It then sends the appropriate command calls to the banking application.

Model:

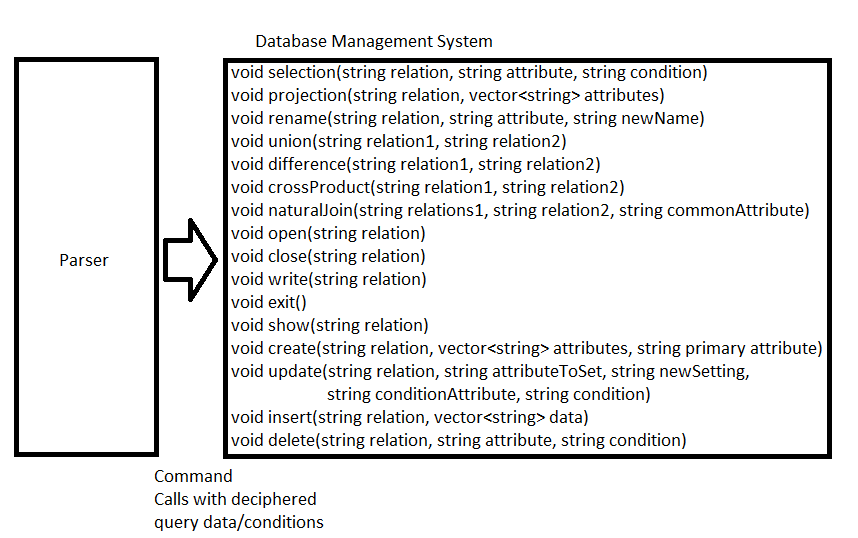
Parser:

Usage: This component is used to convert string queries and commands from the banking application into relational database operations and commands to be executed on the database by the database management system.

Model:

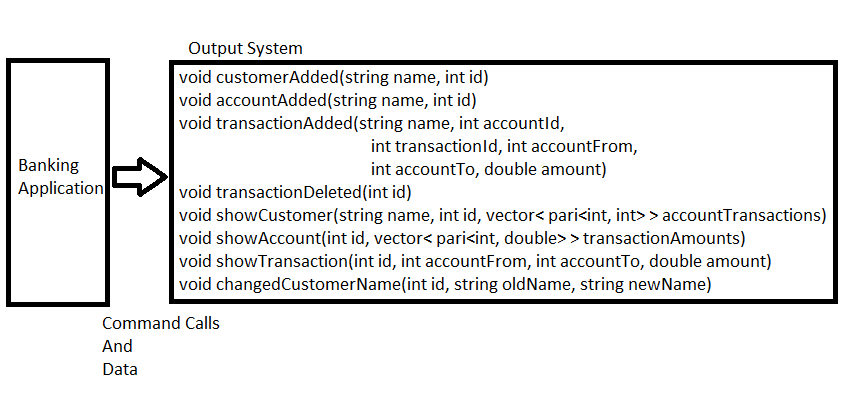
Relationship Database Management System:

Usage: This component executes database operations and commands and returns the resulting data/relations to the banking application for use.

Model:

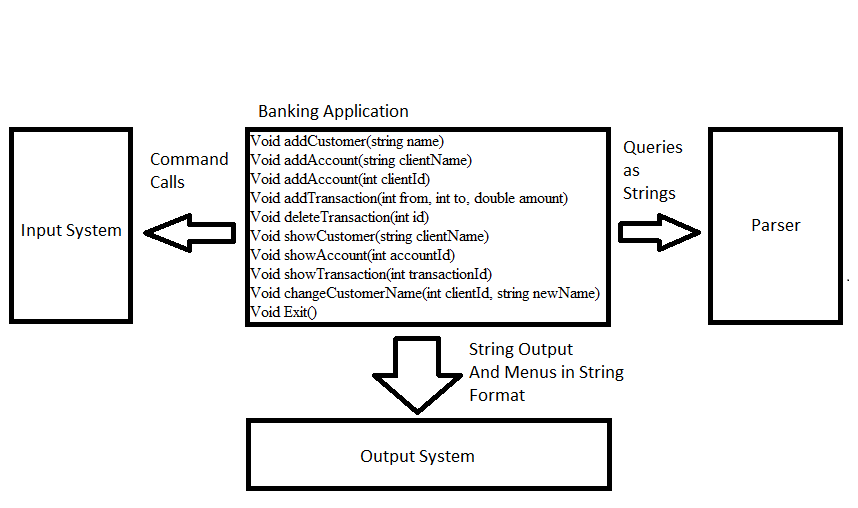
Output Component:

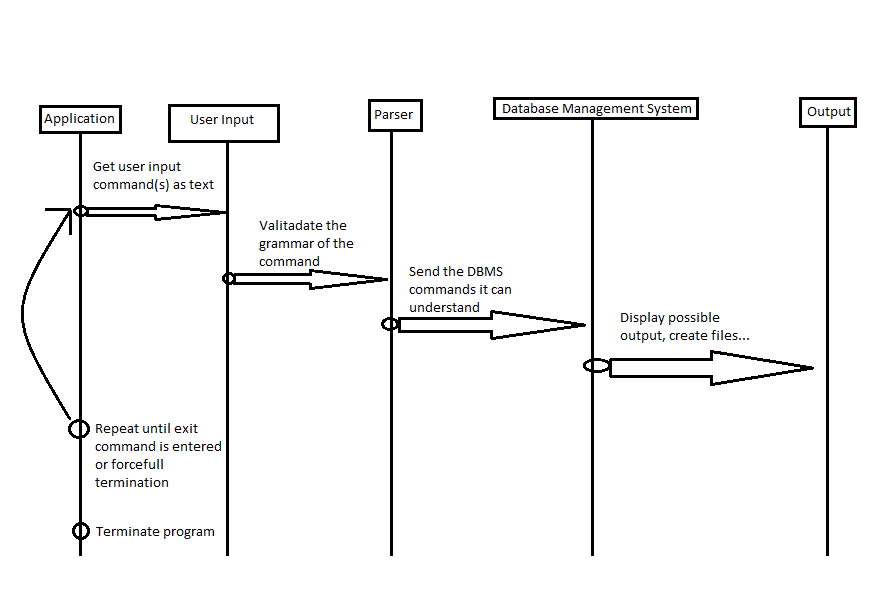
Usage: This component converts data and context supplied by the banking application into string output/menus to be printed to the console.

Model:

Banking Application:

Usage: This component is used by the input system to manipulate the database, via the parser. This component also keeps track of application state and provides data and context to the output system.

Model:

Entity Interactions:

**Section 4 – Benefits, assumptions, risks/issues:**

Benefits:

The database system is very simple, which help building and maintaining it. Its design should allow for composing operations to increase its usefulness and ease of use. The application will help with keeping track of simple money transactions involving any client’s account. Our application automates keeping records, which even careful people are bound to mishandle.

Assumptions:

We are assuming that the simplistic database will be robust enough to meet the needs of our application. We are assuming that the user will be able to view money transaction records easily. We are assuming that we will be able to create a system capable of understanding entities within relations. We are assuming that the database will only be used to store entities built from integers and strings, with no support for Unicode.

Risks:

The data is stored in plaintext. This makes the database insecure and vulnerable to attacks, but adheres to the UNIX principles, and makes using the application and handling the data easier. The app has the potential to waste the users’ time, or be too simple to be of any real use.