**Chapter 3**

**SYSTEM DESIGN**

**System Architecture:**

**Introduction:**

The purpose of the design phase is to plan a solution of the problem specified by the requirements document. This phase is the first step in moving from the problem domain to the solution domain. In other words, starting with what is needed; design takes us toward how to satisfy the needs. The design of a system is perhaps the most critical factor affecting the quality of the software; it has a major impact on the later phases particularly testing and maintenance.

### 

### **FUNCTIONAL SPECIFICATION/DESIGN METHODOLOGY:**

* + - System is used to identifying the most dangerous place in terms of road traffic.
    - System can be used to take necessary actions to improve the road safety.
    - System is automation for road accident prediction and patterns.
    - System contains 3 actors namely administrator, traffic in charger and visitor.
    - System makes use of road accidents constraints such as number of vehicles, speed limit, road type, light conditions, weather conditions, road surface, junction etc.
    - System predicts road accidents and their patterns.
    - System prediction patterns based on the constraints such as city, road name, and month wise etc.

### **BROAD DESIGN:**

According to Software Engineering the approach adopted to develop this project is the Iterative waterfall Model. The iterative waterfall Model is a systematic approach that begins at the feasibility study phase and progress through analysis, design, coding, testing, integration and maintenance. Feedback paths are there in each phase to its preceding phase as show in the figure to allow the correction of the errors committed during a phase that are detected in later phase.

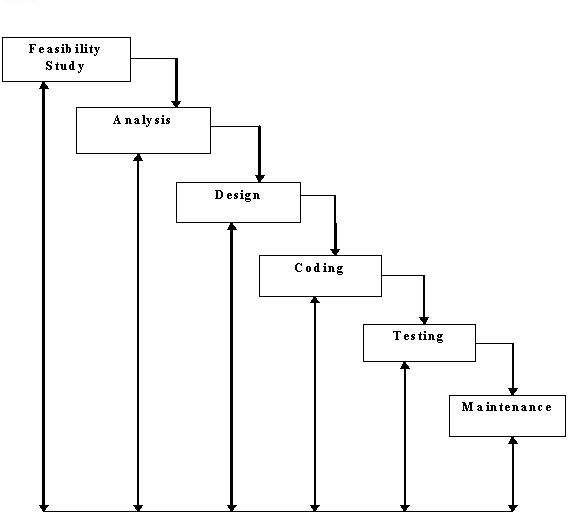


Fig.1: Waterfall model

### 

### **DATA FLOW DIAGRAM:**

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an [information system.](http://en.wikipedia.org/wiki/Information_system) DFDs can also be used for the [visualization](http://en.wikipedia.org/wiki/Data_visualization) of [data](http://en.wikipedia.org/wiki/Data_processing) [processing](http://en.wikipedia.org/wiki/Data_processing) (structured design). On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process. A DFD provides no information about the timing of processes, or about whether processes will operate in sequence or in parallel. It is therefore quite different from a [flowchart,](http://en.wikipedia.org/wiki/Flowchart) which shows the flow of control through an algorithm, allowing a reader to determine what operations will be performed, in what order, and under what circumstances, but not what kinds of data will be input to and output from the system, nor where the data will come from and go to, nor where the data will be stored (all of which are shown on a DFD).

#### **Symbols used in DFD’s:**

1. **Processes:**

A process transforms data values. The lowest processes are our functions without side effects.

#### **Data Flows:**

A data flow connects the output of an object or process to the input of another object or process. It represents the intermediate data values within the computation. It is draws as an arrow between the procedure and the consumer of the data value. The arrow is labeled with the description of the data, usually its name or type.

#### **Actors:**

An actor is an active object that drives the data flow graph by producing or consuming values. Actors are attached to the inputs and the outputs of a dataflow graph. In sense, the actors lie on the boundary of the flow graph but terminate the flow of data as sources and sinks of data, and so are sometimes called terminators.

#### **Data Store:**

A data store is a passive object within a data flow diagram that stores data for later access. Unlike an actor, a data store does not generate any operations on its own but merely responds to requests to store and access data.

#### **Level 1 (high level diagram):**

This level (level 1) shows all processes at the first level of numbering, data stores, external entities and the data flows between them. The purpose of this level is to show the major and high-level processes of the system and their interrelation. A process model will have one, and only one, level-1 diagram. A level-1 diagram must be balanced with its parent context level diagram, i.e. there must be the same external entities and the same data flows, these can be broken down to more detail in the level1.

Figure 1 shows the detailed data flow. Apart from viewing basic details, public doesn’t involve in many functionalities. So, data flow diagram of Use is only shown. A User after successful login is provided with functionalities like viewing upcoming sports events, view mentors for different sports, view shortlisted students for upcoming events and view gallery. Actions performed by the actors are constantly updated to the respective database as shown.

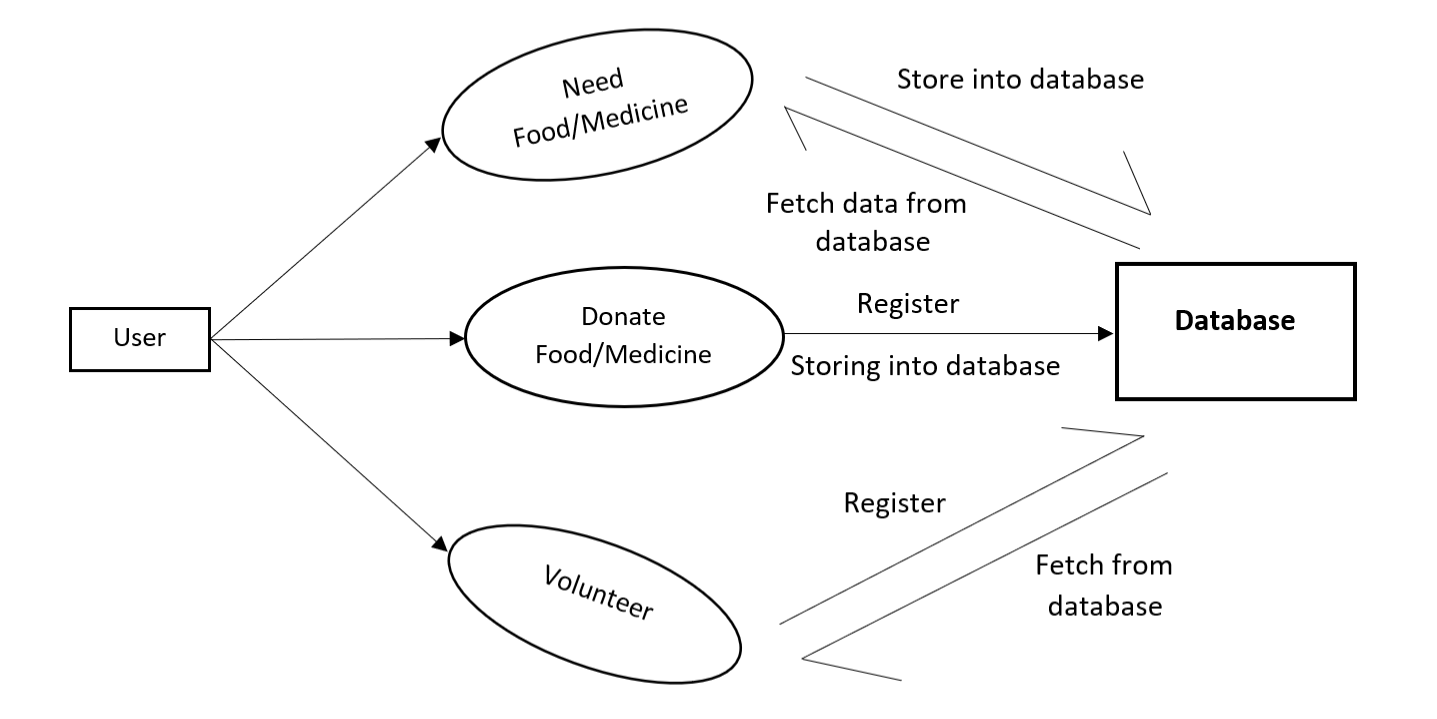


Fig.1: Data flow diagram (User)

### **DETAILED DESIGN:**

#### **Introduction:**

Systems design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development.

#### **Architecture of the System:**

#### **Three tier Architecture:**

Three-tier architecture is a client-server architecture in which the functional process logic, data access, computer data storage and user interface are developed and maintained as independent modules on separate platforms.

#### **The Data Layer:**

The key component to most applications is the data. The data has to be served to the presentation layer somehow. The data layer is a separate component (often setup as a separate single or group of projects in a .NET solution), whose sole purpose is to serve up the data from the database and return it to the caller. Through this approach, data can be logically reused, meaning that a portion of an application reusing the same query can make a call to one data layer method, instead of embedding the query multiple times. This is generally more maintainable.

#### **Business Layer:**

Though a web site could talk to the data access layer directly, it usually goes through another layer called the business layer. The business layer is vital in that it validates the input conditions before calling a method from the data layer. This ensures the data input is correct before proceeding, and can often ensure that the outputs are correct as well. This validation of input is called business rules, meaning the rules that the business layer uses to make “judgments” about the data. One of the best reasons for reusing logic is that applications that start off small usually grow in functionality. The business layer helps move logic to a central layer for “maximum reusability.”

#### **Presentation Layer:**

The ASP.NET web site or windows forms application (the UI for the project) is called the presentation layer. The presentation layer is the most important layer simply because it’s the one that everyone sees and uses. Even with a well-structured business and data layer, if the presentation layer is designed poorly, this gives the users a poor view of the [system.](http://dotnetslackers.com/articles/net/IntroductionTo3TierArchitecture.aspx#%23)

The **presentation tier** contains the UI (User Interface) elements of the site, and includes all the logic that managers the interaction between the visitor and the client’s business. (ASP.NET Web Forms, Web User Controls, ASP.NET Master Pages)

The **business tier** receives requests from the presentation tier and returns a result to the presentation tier depending on the business logic it contains. (C# Classes)

The **data tier** is responsible for storing the application’s data and sending it to the business tier when requested. (SQL Server Stored Procedures).

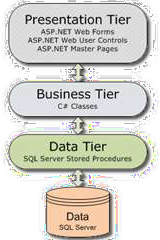


Fig.5: Three tier Architecture**.**

### 

### **MODULE DESIGN:**

#### This application mainly consists of 2 actors:

1. **Administrator**

Administrator is a one who maintains the entire system. Administrator is responsible for updating all information regarding upcoming sports events and students’ information.

#### **User (Public)**

User visits the application. Visitor has limited accessibility.

### **DATABASE DESIGN:**

**Database:**

**MongoDB** is a document-oriented NoSQL database used for high volume data storage. Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents. Documents consist of key-value pairs which are the basic unit of data in MongoDB. Collections contain sets of documents and function which is the equivalent of relational database tables. MongoDB is a database which came into light around the mid-2000s.

MongoDB is a **distributed database at its core**, so high availability, horizontal scaling, and geographic distribution are built in and easy and free to use. It comes with the scalability and flexibility that you want with the querying and indexing that you need. MongoDB **stores data in flexible, JSON-like documents**, meaning fields can vary from document to document and data structure can be changed over time. The document model **maps to the objects in your application code**, making data easy to work with. **Ad hoc queries, indexing, and real time aggregation** provide powerful ways to access and analyse your data.

The data model available within MongoDB allows you to represent hierarchical relationships, to store arrays, and other more complex structures more easily. MongoDB uses the concept of shading to scale horizontally by splitting data across multiple MongoDB instances. MongoDB can run over multiple servers, balancing the load and/or duplicating data to keep the system up and running in case of hardware failure.

In this project, we have used mainly five collections as shown in the Schema Design. We’ve created a collection for storing food-donor details, medicine-donor details, food-receiver details, medicine-receiver details and volunteer details. The Schema Design in the next section will provide a better view.

**Schema Design:**

MongoDB schema design is the most critical part of deploying a scalable, fast and affordable database. It works differently than relational schema design. It provides two types of data models: — Embedded data model and Normalized data model. Based on the requirement, you can use either of the models while preparing your document. While designing schema, one should consider good query performance and reasonable amount of hardware. The following figure shows Schema Design of our project.

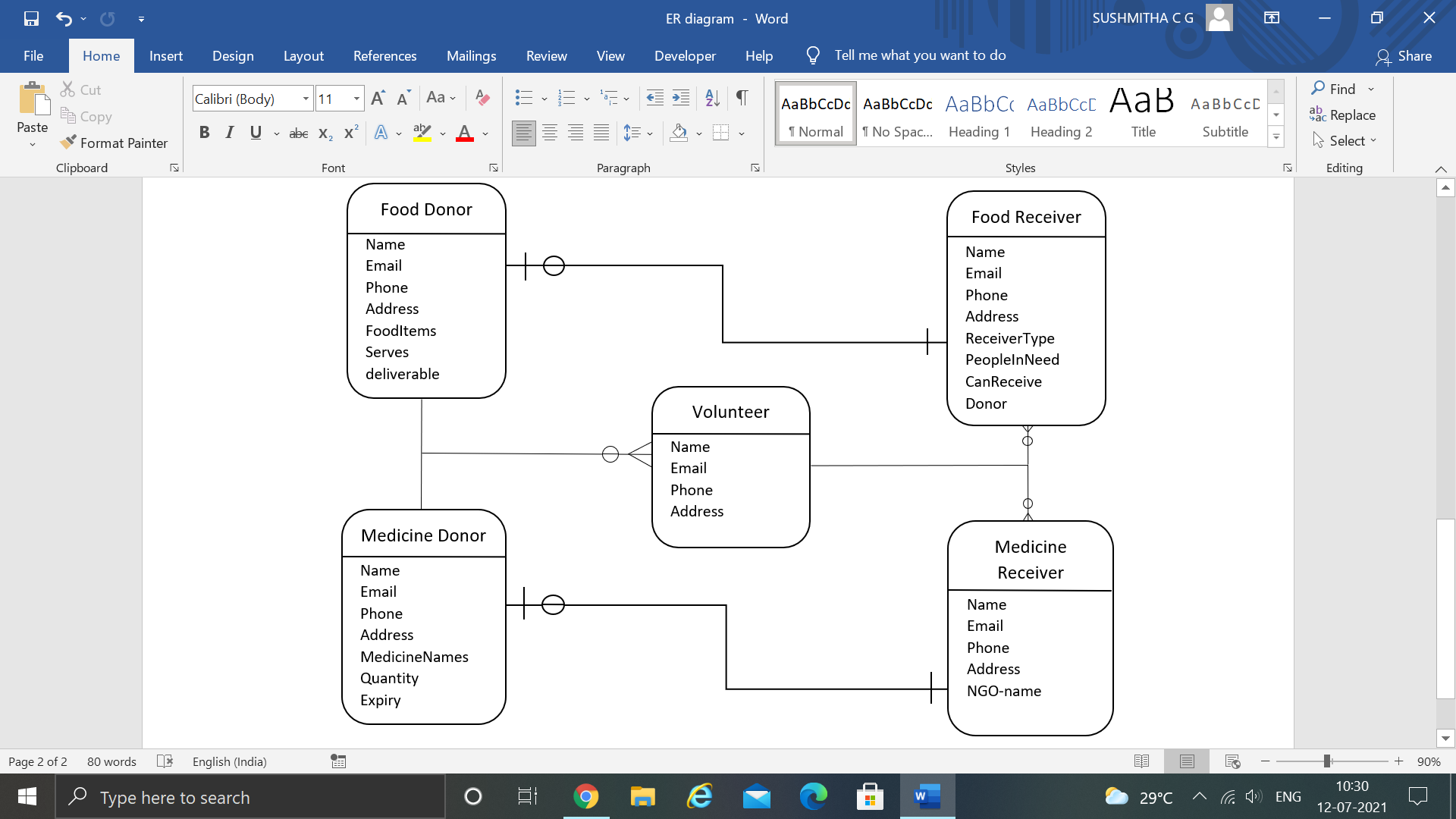


Fig.7: Schema Design