**3. System Design**

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.

**3.1 Database Design**

Database Designing is a part of the development process. In the linear development cycle, it is used during the system requirements phase to construct the data components of the analysis model. This model represents the major data objects and the relationship between them. It should not be confused with data analysis, which takes place in the system design phase. As in a DFD, a model of data consists of a number of symbols joined up according to certain conventions.

**Introduction**

The data pertaining to proposed systems are voluminous that careful design of the database must proceed storing the data in the database. A database management system provides flexibility in storing and retrieving of data and production of information. The RDBMS is a needed and how they are processed and the operating system of the computer, which is responsible for placing database that a specific program will use.

**3.2 Entity Relationship Diagram**

The Entity relationship analysis uses three major abstractions to describe data. These are

* Entities, which are distinct things in the enterprise.
* Relationships, which are meaningful interactions between the objects.
* Attributes, which are the properties of the entities and relationships.

The relative simplicity and pictorial clarity of this diagramming technique may well account in large part for the widespread use of ER model.

**E-R Diagram Components**

Rectangles, which represent the entity set.

Ellipse, which represent attributes set.

Diamonds, which represent relationship sets

Lines, which link attributes to entity sets and

Entity sets to relationships.

Double ellipse which represents multi-valued attributes

**E-R Diagram**

**Registration**

**Carrying**

**Login**

**Contains**

**Import data set**

**Contains**

**Accuracy**

**Figure 3.2.1:** E-R Diagram for HPPQ

**3.3 Dataflow Diagrams**

**DFD SYMBOLS**

Square defines a source (originator) or destination of system data. An arrow identifies data flow. It is the pipeline through which the information flows. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows. 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

**SAILENT FEATURES OF DFD’S**

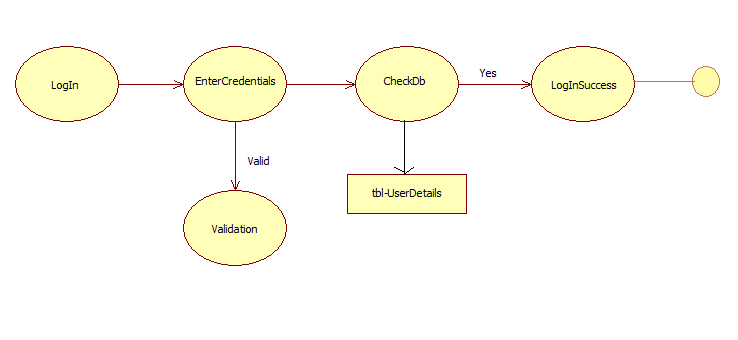
* The DFD shows flow of data, not of control loops and decision are controlled considerations do not appear on a DFD.
* The DFD does not indicate the time factor involved in any process whether the dataflow take place daily, weekly, monthly or yearly.
* The sequence of events is not brought out on the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

**Registration Dataflow Diagrams**



**Figure 3.3.1:** Registration for Dataflow Diagram

**Login Dataflow Diagram**



**Figure3.3.2 :** Login For Dataflow Diagram

**User Data Flow Diagram**

registation

Eating dataset

Eating dataset

Eating dataseet

Eating dataset

**Figure 3.3.3:** User Dataflow Diagram

**3.4 Data Dictionary**

Data dictionary consists of descriptions of all the data used in the system. It consists of logical characteristic of current system data stores, including name, description, aliases, contents and organization. Data dictionary serves as a basis for identifying database requirements during requirements during system design. Data dictionary is a catalog, a repository of the elements in the system.

**Table Name: Registration**

**Description:** This table contains the all the details of the users who are registered. For registering the user can must fill all the fields which are specified in the Registration page.

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Data type** | **Size** | **Constraint** |
| First name | Varchar2 | 30 | Not null |
| Last name | Varchar2 | 20 | Not null |
| User name | Varchar2 | 20 | Primary key |
| password | Varchar2 | 10 | Not null |
| email\_id | Varchar2 | 30 | not null |
| mobile no | number | 10 | not null |
| profile | Varchar2 | 40 | not null |

**Table 3.4.1:** Registration

**Table Name: Login**

**Description:** This table contains username, password. User enters the correct details.

|  |  |  |  |
| --- | --- | --- | --- |
| **Column name** | **Datatype** | **Size** | **Constraint** |
| username | Varchar2 | 20 | Primary key |
| Password | Varchar2 | 10 | not null |

**Table 3.4.2**: Login

**3.5 UML Diagrams**

The Unified Modeling Language (UML) is a visual modeling language used to specify, visualize, construct and document a software intensive system. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process.

**Building Blocks of the UML**

The vocabulary of the UML encompasses three kinds of building blocks

1. Things

2. Relationships

3. Diagrams

Things are the abstractions that are first class citizens in a model, relationships tie these things together and diagram group interesting collections of things.

**Things in the UML**

There are four kinds of things in the UML

1. Structural Things
2. Behavioral Things
3. Grouping Things
4. Annotational Things

These things are the basic object-oriented building blocks of the UML, you can use them to write well-formed models.

**Structural things**

Structural things are the nouns of UML models. These are, the more static parts of a model, representing elements that are either conceptual or physical. In all there are seven kinds of structural things.

**Classes**

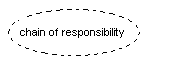
First, a class is a description of a set of objects that share the same attributes, operations, relationships and semantics. A class implements one or more interfaces graphically, a class is rendered as a rectangle, usually including its, name, attributes, and operations.

|  |
| --- |
| Windows |
| Origin  Size |
| Open()  Close()  Move()  Display() |

**Figure:** Sample Class Diagram

**Collaborations**

Third, collaboration defines an interaction and is a society of roles and other elements that work together to provide some cooperative behavior that’s bigger than the sum of all the elements. Therefore, collaborations have structural, as well as behavioral, dimensions.



**Figure:** Sample Collaboration Diagram

**Active Classes**

Fifth, an active class is a class whose objects own one or more processes or threads and therefore can initiate control activity. Graphically, an active class is rendered just like a class, but with heavy lines, usually including its name, attributes, and operations. They represent physical things, whereas the previous five things represent conceptual or things.

|  |
| --- |
| Event Manager |
|  |
| Suspend()  Flush() |

**Figure:** Sample Active Class Diagram

**Behavioral Things**

Behavioral things are the dynamic parts of UML models. These are the verbs of a model representing behavior over time and space. In all, there are two primary kinds of behavioral things. First, an interaction is a behavior that comprises and links graphically message is rendered as a directed line, almost always including the name of its operation.

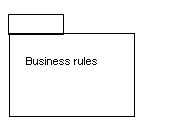
**Figure:** Directed Line

**Grouping Things**

Grouping things are the original parts of UML models. These are the boxes into which a model can be decomposed. In all, there is one primary kind of grouping thing, namely, packages.

**Packages**

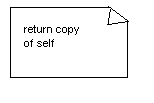
A package is a general-purpose mechanism for organizing elements into groups. Structural things, behavioral things, and even other grouping things may be placed in a package. Unlike components (which exist at run time), a package is purely conceptual (meaning that exist only development time). Graphically, a package is rendered as a tabbed folder, usually including only its name and, sometimes, its contents.



**Figure:** Sample Packages Diagram

**Annotational Things**

Annotational things are the explanatory parts of UML model. These are the comments you may apply to describe, illuminate, and remark about any element in a model.



**Figure:** Sample Anchor Note Diagram

**Relationships in the UML**

There are four kinds of relationships in the UML.

1. Dependency

2. Association

3. Generalization

4. Realization

First, a dependency is a semantic relationship between two things in which a change to one thing the independent thing) may affect the semantics of the other thing (the dependent thing). Graphically dependency is rendered as a dashed line, possibly directed, and occasionally including a label.

Dependency

Second, a generalization is a specialization/generalization relationship in which objects of the specialized element (the child) are substitutable for objects of the generalized element (the parent).

Generalization

Third, a realization is a semantic relationship between classifiers, wherein one classifier specifies a contract that other classifier guarantees to carry out. Graphically, a realization relationship is rendered as a cross between a generalization and a dependency relationship.

Realization

**Diagrams in the UML**

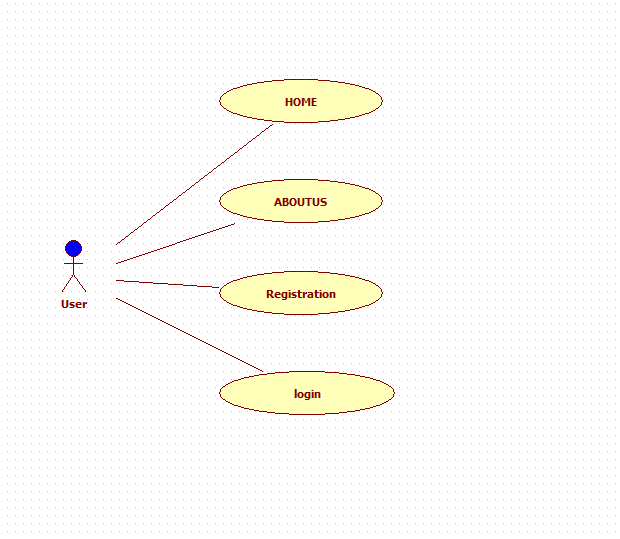
A diagram is the graphical presentation of a set of elements most often rendered as a connected graph of vertices (things) and arcs (relationships). You draw diagrams to visualize a System from different perspectives, so a diagram represents an elided view of the elements that make up a System. The same element may appear in all diagrams, which are consistent with the five most useful views that comprise the architecture of a software intensive System.

UML includes nine such diagrams.

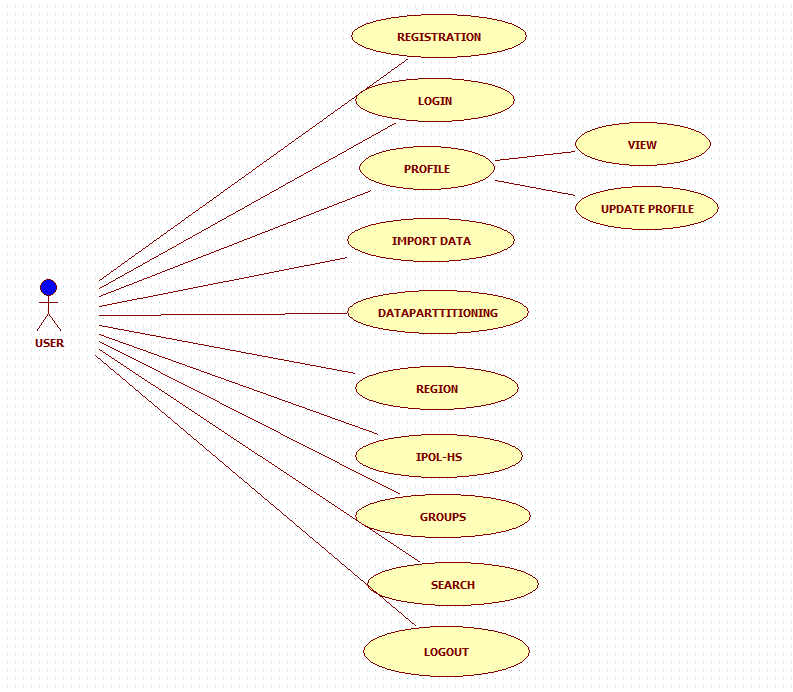
* Object diagram.
* Class diagram.
* Use case diagram.
* Sequence diagram.
* Collaboration diagram.
* Deployment diagram.

**Construction of Use case diagrams**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases).

****

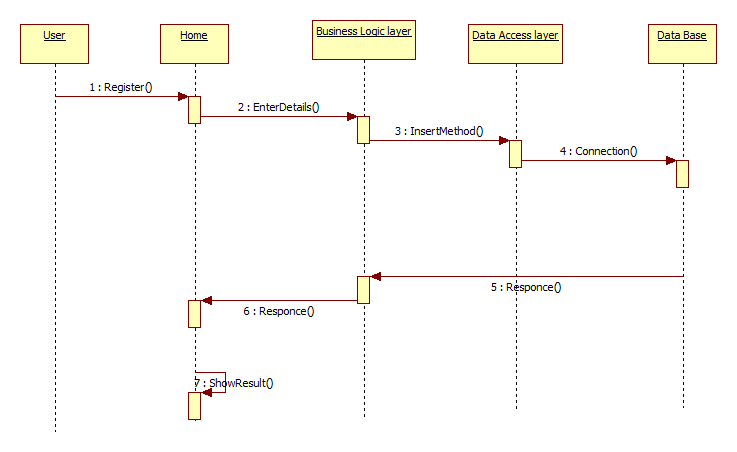
**Figure 3.5.1:** Use case diagram for Home page.

****

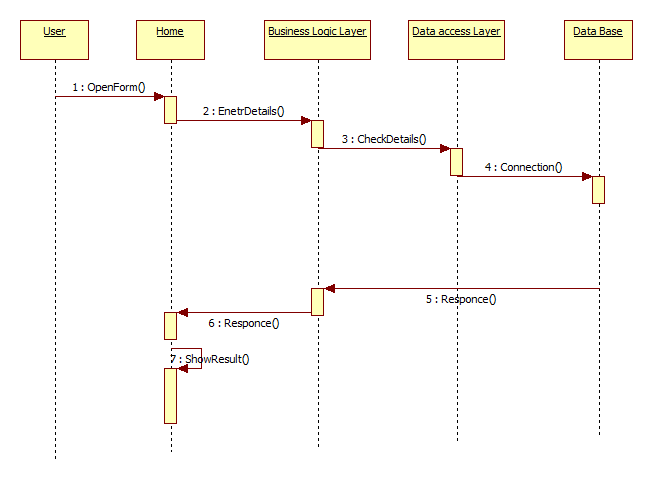
**Figure 3.5.2:** Use case diagram for User.

**Sequence Diagram**

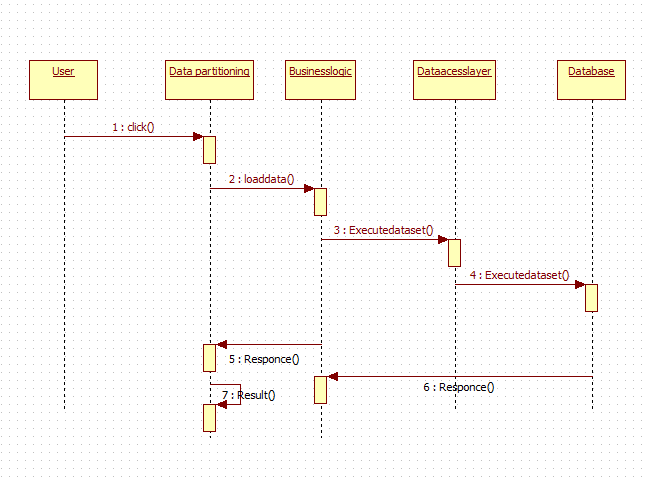
A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

****

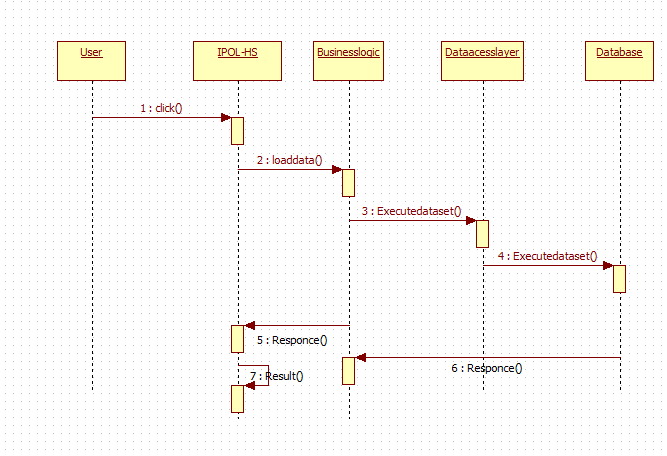
**Figure 3.5.3:** Sequence diagram for Registration.



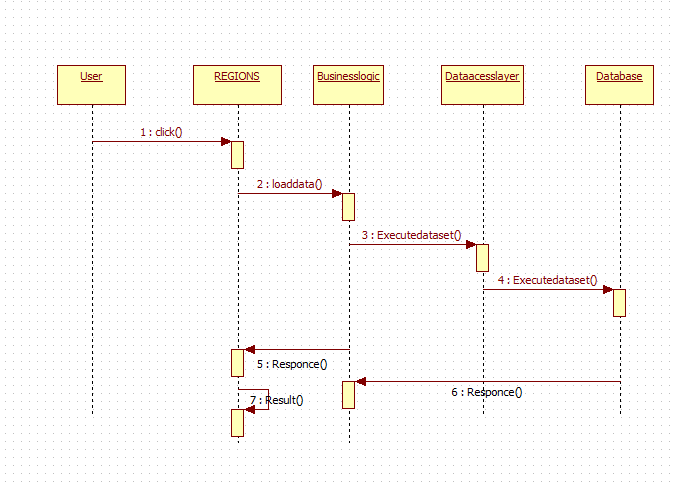
**Figure 3.5.4:** Sequence diagram for Login.

****

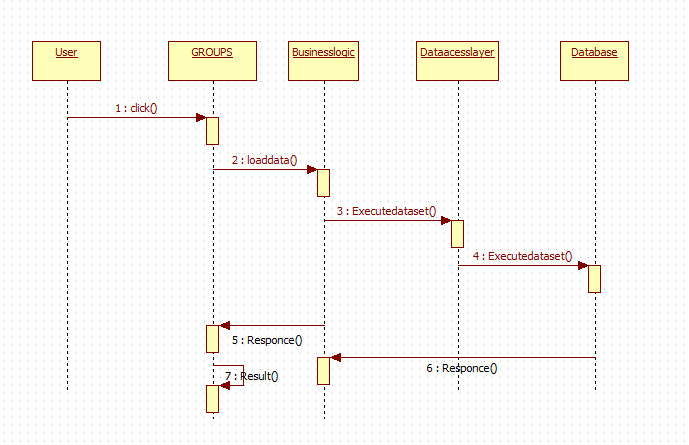
**Figure 3.5.5:** Sequence diagram for Data Partitioning.

****

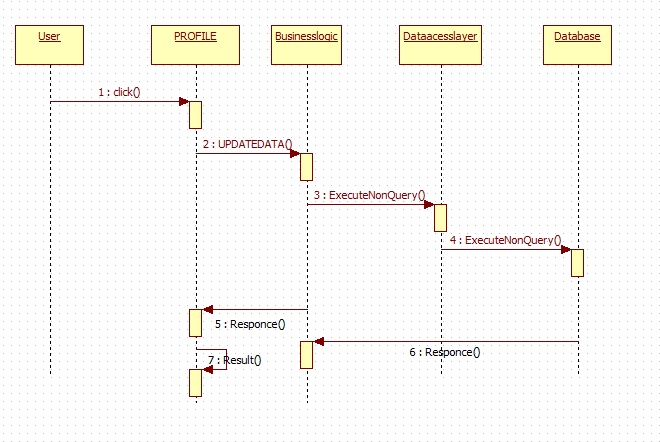
**Figure 3.5.6:** Sequence diagram for IPOL-HS.

****

**Figure 3.5.7:** Sequence diagram for Regions.



**Figure 3.5.8:** Sequence diagram for Groups.



**Figure 3.5.9:** Sequence diagram for Update Profile.

**Class Diagram**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



**Figure 3.5.10:** Class diagram for whole system.

**Activity Diagram**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system.

Get the Details

Validate Data

Invalid

Accept

Enter User Registration Details

Submit

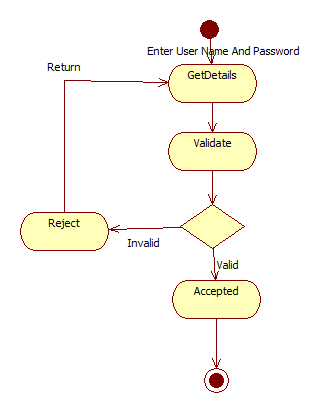
No

Yes

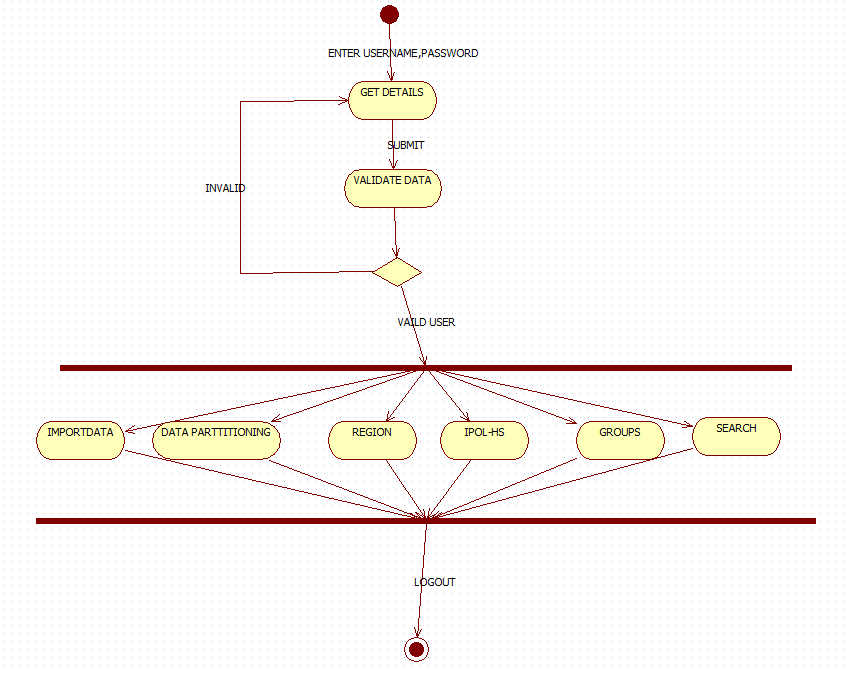
Returns Error Message

Successfully Registered

**Figure: 3.5.11:** Activity diagram for Registration.

****

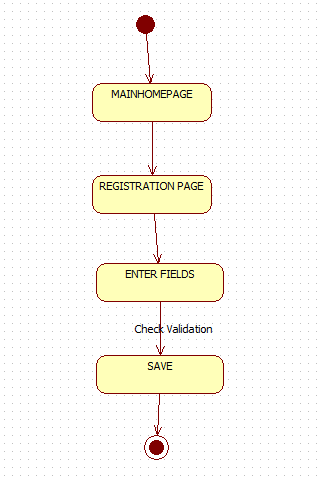
**Figure: 3.5.12:** Activity diagram for Login.



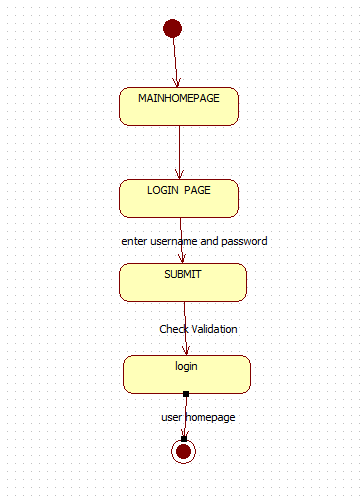
**Figure: 3.5.13:** Activity diagram for User.

**State chart diagram**

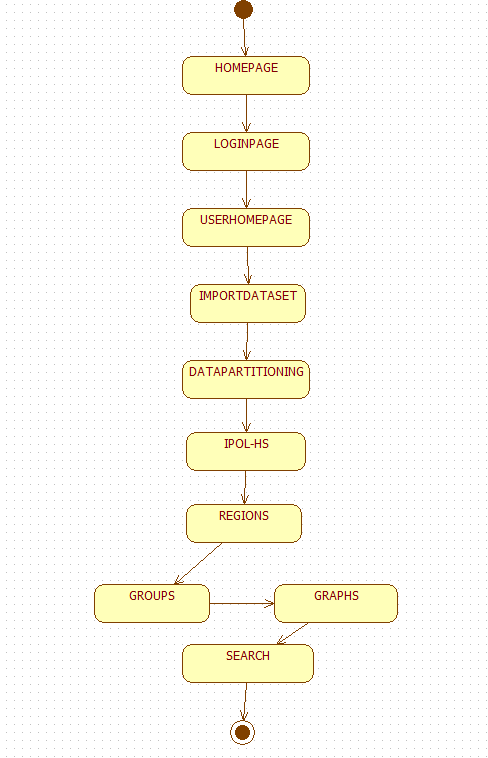
State chart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. State chart diagrams are useful to model the reactive systems. State chart diagram describes the flow of control from one state to another state.



**Figure 3.5.14:** State chart diagram for Registration.



**Figure 3.5.15:** State chart diagram for Login.



**Figure 3.5.16:** State chart diagram for User.