**Quick Closest Neighbor Search with Keywords**

**Abstract:**

Conventional spatial queries, such as range search and nearest neighbor retrieval, involve only conditions on objects’ geometric properties. Today, many modern applications call for novel forms of queries that aim to find objects satisfying both a spatial predicate, and a predicate on their associated texts. For example, instead of considering all the restaurants, a nearest neighbor query would instead ask for the restaurant that is the closest among those whose menus contain “steak, spaghetti, brandy” all at the same time. Currently the best solution to such queries is based on *the IR*2*-tree*, which, as shown in this paper, has a few deficiencies that seriously impact its efficiency. Motivated by this, we develop a new access method called *the spatial inverted index* that extends the conventional inverted index to cope with multidimensional data, and comes with algorithms that can answer nearest neighbor queries with keywords in real time. As verified by experiments, the proposed techniques outperform the IR2-tree in query response time significantly, often by a factor of orders of magnitude.

INTRODUCTION

A spatial database manages multidimensional objects (such as points, rectangles, etc.), and provides fast access to those objects based on different selection criteria. The importance of spatial databases is reﬂected by the convenience of modeling entities of reality in a geometric manner. For example, locations of restaurants, hotels, hospitals and so on are often represented as points in a map, while larger extents such as parks, lakes, and landscapes often as a combination of rectangles. Many functionalities of a spatial database are useful in various ways in speciﬁc contexts. For instance, in a geography information system, range search can be deployed to ﬁnd all restaurants in a certain area, while nearest neighbor retrieval can discover the restaurant closest to a given address. Today, the widespread use of search engines has made it realistic to write spatial queries in a brandnew way. Conventionally, queries focus on objects’ geometric properties only, such as whether a point is in a rectangle, or how close two points are from each other. We have seen some modern applications that call for the ability to select objects based on both of their geometric coordinates and their associated texts. For example, it would be fairly useful if a search engine can be used to ﬁnd the nearest restaurant that offers “steak, spaghetti, and brandy” all at the same time. Note that this is not the “globally” nearest restaurant (which would have been returned by a traditional nearest neighbor query), but the nearest restaurant among only those providing all the demanded foods and drinks.

In this paper, we design a variant of inverted index that is optimized for multidimensional points, and is thus named the spatial inverted index (SI-index). This access method successfully incorporates point coordi- nates into a conventional inverted index with small extra space, owing to a delicate compact storage scheme. Meanwhile, an SI-index preserves the spatial locality of data points, and comes with an R-tree built on every inverted list at little space overhead. As a result, it offers two competing ways for query processing. We can (sequentially) merge multiple lists very much like merging traditional inverted lists by ids. Alternatively, we can also leverage the R-trees to browse the points of all relevant lists in ascending order of their distances to the query point. As demonstrated by experiments, the SI-index signiﬁcantly outperforms the IR 2 -tree in query efﬁciency, often by a factor of orders of magnitude.

**EXISTING SYSTEM:**

Spatial queries with keywords have not been ex-tensively explored. In the past years, the community has sparked enthusiasm in studying keyword search in relational databases. It is until recently that attention was diverted to multidimensional data. Existing works mainly focus on finding top-k Nearest Neighbours, where each node has to match the whole querying keywords .It does not consider the density of data objects in the spatial space. Also these methods are low efficient for incremental query.

We have ﬁnished explaining how to build the leaf nodes of an R-tree on an inverted list. As each leaf is a block, all the leaves can be stored in a blocked SI-index Building the nonleaf levels is trivial, because they are invisible to the merging-based query algorithms, and hence, do not need to preserve any common ordering. We are free to apply any of the existing R-tree construction algorithms. It is noteworthy that the nonleaf levels add only a small amount to the overall space overhead because, in an R-tree, the number of nonleaf nodes is by far lower than that of leaf nodes.

**PROPOSED SYSTEM:**

A spatial database manages multidimensional objects (such as points, rectangles, etc.), and provides fast access to those objects based on different selection criteria. The importance of spatial databases is reflected by the convenience of modeling entities of reality in a geometric manner. For example, locations of restaurants, hotels, hospitals and so on are often represented as points in a map, while larger extents such as parks, lakes, and landscapes often as a combination of rectangles. Many functionalities of a spatial database are useful in various ways in specific contexts. For instance, in a geography information system, range search can be deployed to find all restaurants in a certain area, while nearest neighbor retrieval can discover the restaurant closest to a given address.

Furthermore, as the SI-index is based on the conventional technology of inverted index, it is readily incorporable in a commercial search engine that appliesmassive parallelism, implying its immediate industrialmerits.

**Modules :**

1. **Registration**
2. **Login**
3. **Hotel\_Registration**
4. **Search Techniques**
5. **Map\_view**
6. **Distance\_Search**

**Modules Description**

**Registration:**

In this module an User have to register first,then only he/she has to access the data base.

**Login:**

In this module,any of the above mentioned person have to login,they should login by giving their email id and password .

**Hotel\_Registration:**

In this module Admin registers the hotel along with its famous dish.Also he measures the distance of the corresponding hotel from the corresponding source place by using spatial distance of Google map

**Search Techniques:**

Here we are using two techniques for searching the document 1)Restaurant Search,2)Key Search.

Key Search:

It means that the user can give the key in which dish that the restaurant is famous for .This results in the list of menu items displayed.

Restaurant Search:

It means that the user can have the list of restaurants which are located very near. List came from the database.

**Map\_View:**

The User can see the view of their locality by Google Map(such as map view, satellite view) .

**Distance\_Search:**

The User can measure the distance and calculate time that takes them to reach the destination by giving speed. Chart will be prepared by using these values. These are done by the use of Google Maps.

**TECHNOLOGIES USED**

The following depicts the program that runs on java platform.

**Functional and Non-Functional Requirements:**

1. **Functoinal Requirements:**
2. **Inputs:**

Browsing and uploading of files.

1. **Processing:**

Cluster server: There are 3 cluster servers Cluster server1 stores files of server1.Cluster server2 stores files of server2.cluster server3 stores files of server3.

Load server: Stores all files

Slip server cluster:

* Browses the file
* Selects the path
* Download the fie

**Output:** SIP user agent clients select file and location to download the file. To download the selected file server will send file to the SIP user agent.

1. **Non Functional Requirements**

Performance is measured in terms of the output provided by the application.

The requirement specification for any system can be broadly stated as given below:

* The system should be able to interface with the existing system.
* The system should be accurate.
* Te system should be better than existing system.

**Portability**

* **Reliability**
* **Reusability**
* **Robustness**
* **Testability**
* **Usability**
* **Security**

**DESIGN ANALYSIS**

**UML Diagrams:**

UML is a method for describing the system architecture in detail using the blueprint.

UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

UML is a very important part of developing objects oriented software and the software development process.

UML uses mostly graphical notations to express the design of software projects.

Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

**Definition:**

UML is a general-purpose visual modeling language that is used to specify, visualize, construct, and document the artifacts of the software system.

**UML is a language:**

It will provide vocabulary and rules for communications and function on conceptual and physical representation. So it is modeling language.

**UML Specifying:**

Specifying means building models that are precise, unambiguous and complete. In particular, the UML address the specification of all the important analysis, design and implementation decisions that must be made in developing and displaying a software intensive system.

**UML Visualization:**

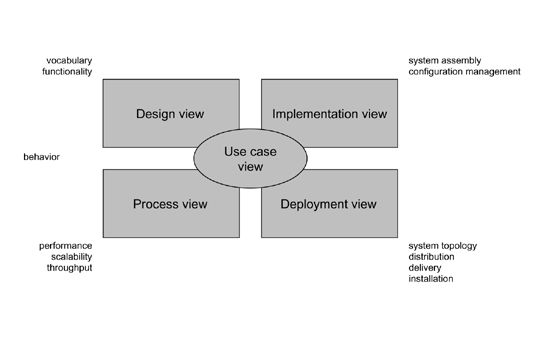
The UML includes both graphical and textual representation. It makes easy to visualize the system and for better understanding.

**UML Constructing:**

UML models can be directly connected to a variety of programming languages and it is sufficiently expressive and free from any ambiguity to permit the direct execution of models.

**UML Documenting:**

UML provides variety of documents in addition raw executable codes.



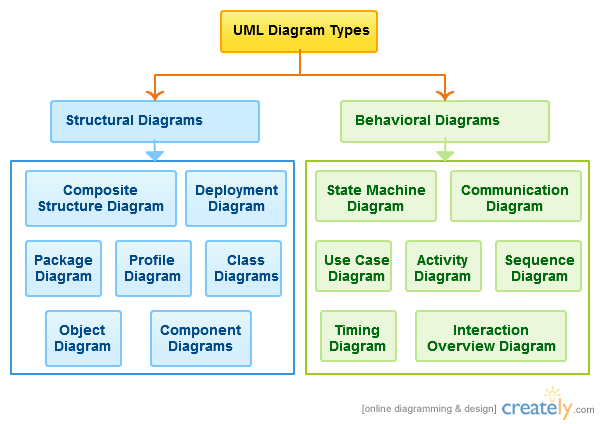
**Modeling a System Architecture using views of UML**

The use case view of a system encompasses the use cases that describe the behavior of the system as seen by its end users, analysts, and testers.

The *design view* of a system encompasses the classes, interfaces, and collaborations that form the vocabulary of the problem and its solution.

The *process view* of a system encompasses the threads and processes that form the system's concurrency and synchronization mechanisms.

The *implementation view* of a system encompasses the components and files that are used to assemble and release the physical system.The*deployment view* of a system encompasses the nodes that form the system's hardware topology on which the system executes.



**Uses of UML :**

The UML is intended primarily for software intensive systems. It has been used effectively for such domain as

Enterprise Information System

Banking and Financial Services

Telecommunications

Transportation

Defense/Aerospace

Retails

Medical Electronics

Scientific Fields

Distributed Web

**Building blocks of UML:**

The vocabulary of the UML encompasses 3 kinds of building blocks

Things

Relationships

Diagrams

**Things:**

Things are the data abstractions that are first class citizens in a model. Things are of 4 types

Structural Things, Behavioral Things ,Grouping Things, An notational Things

**Relationships:**

Relationships tie the things together. Relationships in the UML are

Dependency, Association, Generalization, Specialization

**UML Diagrams:**

A diagram is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices (things) and arcs (relationships).

There are two types of diagrams, they are:

Structural and Behavioral Diagrams

**Structural Diagrams:-**

The UML‘s four structural diagrams exist to visualize, specify, construct and document the static aspects of a system. icon View the static parts of a system using one of the following diagrams. Structural diagrams consists of Class Diagram, Object Diagram, Component Diagram, Deployment Diagram.

**Behavioral Diagrams :**

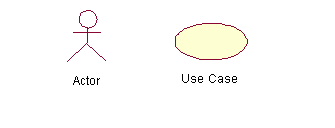
The UML’s five behavioral diagrams are used to visualize, specify, construct, and document the dynamic aspects of a system. The UML’s behavioral diagrams are roughly organized around the major ways which can model the dynamics of a system.

Behavioral diagrams consists of

Use case Diagram, Sequence Diagram, Collaboration Diagram, State chart Diagram, Activity Diagram

**Use-Case diagram:**

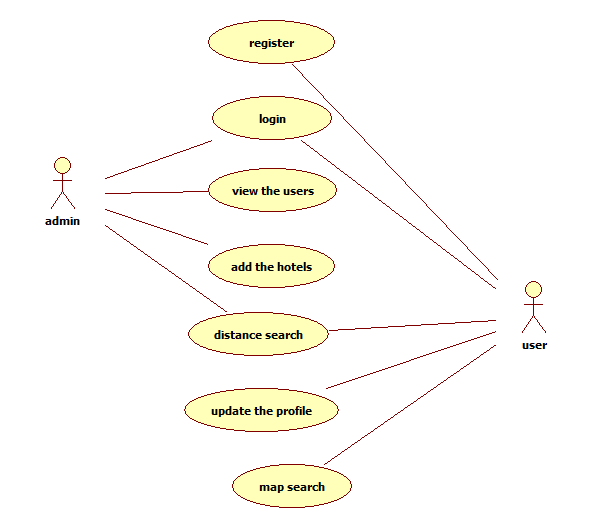
A use case is a set of scenarios that describing an interaction between a user and a system.  A use case diagram displays the relationship among actors and use cases.  The two main components of a use case diagram are use cases and actors.



An actor is represents a user or another system that will interact with the system you are modeling.  A use case is an external view of the system that represents some action the user might perform in order to complete a task.

**Contents:**

* Use cases
* Actors
* Dependency, Generalization, and association relationships
* System boundary



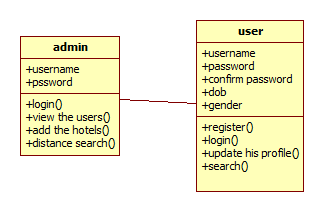
**.**

link for more class diagram examples.

UML Class Diagram with Relationships

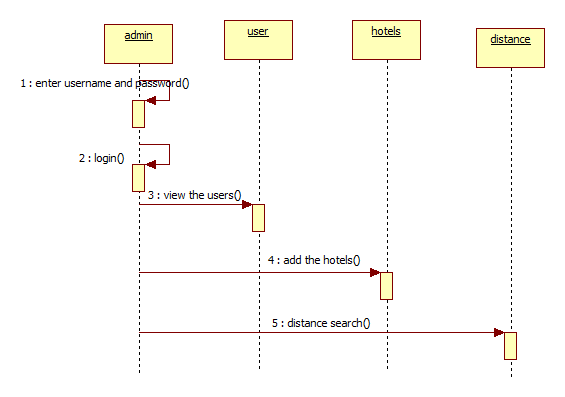
**Class Diagram:**

Class diagrams are widely used to describe the types of objects in a system and their relationships. Class diagrams model class structure and contents using design elements such as classes, packages and objects. Class diagrams describe three different perspectives when designing a system, conceptual, specification, and implementation. These perspectives become evident as the diagram is created and help solidify the design. Class diagrams are arguably the most used UML diagram type. It is the main building block of any object oriented solution. It shows the classes in a system, attributes and operations of each class and the relationship between each class. In most modeling tools a class has three parts, name at the top, attributes in the middle and operations or methods at the bottom. In large systems with many classes related classes are grouped together to to create class diagrams. Different relationships between diagrams are show by different types of Arrows. Below is a image of a class diagram. Follow the scenario

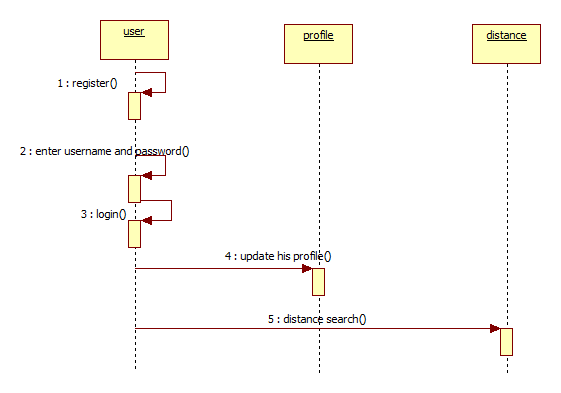


**Sequence diagram**

The processes are represented vertically and interactions are show as arrows. This article explains thepurpose and the basics of Sequence diagrams.

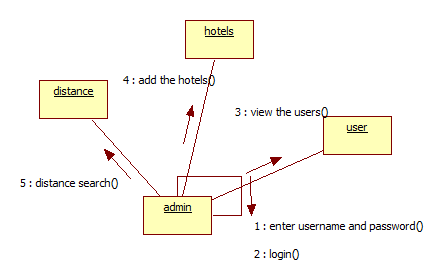


User:

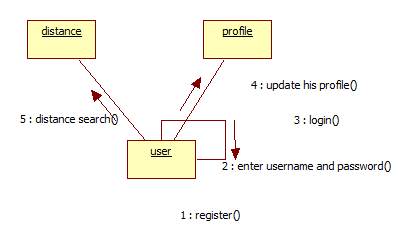


**Collaboration diagram**

Communication diagram was called collaboration diagram in UML 1. It is similar to sequence diagrams but the focus is on messages passed between objects. The same information can be represented using a sequence diagram and different objects. Click here to understand the differences using an example.



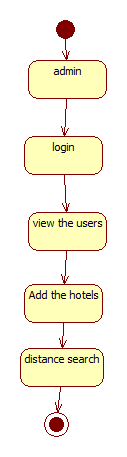
User:



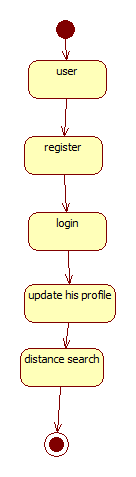
**State machine diagrams**

State machine diagrams are similar to activity diagrams although notations and usage changes a bit. They are sometime known as state diagrams or start chart diagrams as well. These are very useful to describe the behavior of objects that act different according to the state they are at the moment. Below State machine diagram show the basic states and actions.

**Admin:**



User:



**Activity Diagram:**

Activity diagrams describe the workflow behavior of a system.  Activity diagrams are similar to state diagrams because activities are the state of doing something.  The diagrams describe the state of activities by showing the sequence of activities performed.  Activity diagrams can show activities that are conditional or parallel.

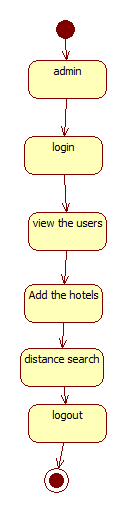
### How to Draw: Activity Diagrams

Activity diagrams show the flow of activities through the system.  Diagrams are read from top to bottom and have branches and forks to describe conditions and parallel activities.  A fork is used when multiple activities are occurring at the same time.  The diagram below shows a fork after activity1.  This indicates that both activity2 and activity3 are occurring at the same time.  After activity2 there is a branch.  The branch describes what activities will take place based on a set of conditions.  All branches at some point are followed by a merge to indicate the end of the conditional behavior started by that branch.   After the merge all of the parallel activities must be combined by a join before transitioning into the final activity state.   .

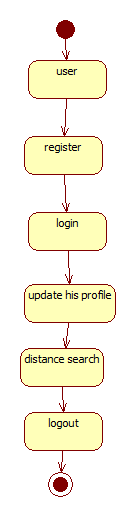
### When to Use: Activity Diagrams

Activity diagrams should be used in conjunction with other modeling techniques such as interaction diagrams and state diagrams.  The main reason to use activity diagrams is to model the workflow behind the system being designed.  Activity Diagrams are also useful for: analyzing a use case by describing what actions need to take place and when they should occur; describing a complicated sequential algorithm; and modeling applications with parallel processes.

Admin:

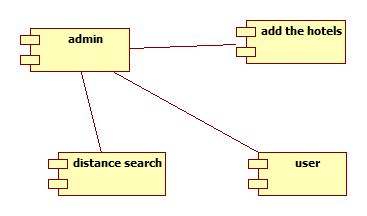


User:



**Component diagram**

A component diagram displays the structural relationship of components of a software system. These are mostly used when working with complex systems that has many components. Components communicate with each other using interfaces. The interfaces are linked using connectors. Below images shows a component diagram.



**Deployment Diagram**

A deployment diagrams shows the hardware of your system and the software in those hardware. Deployment diagrams are useful when your software solution is deployed across multiple machines with each having a unique configuration. Below is an example deployment diagram.



**Data Flow Diagrams (DFD):**

**Admin:**

**Data base**

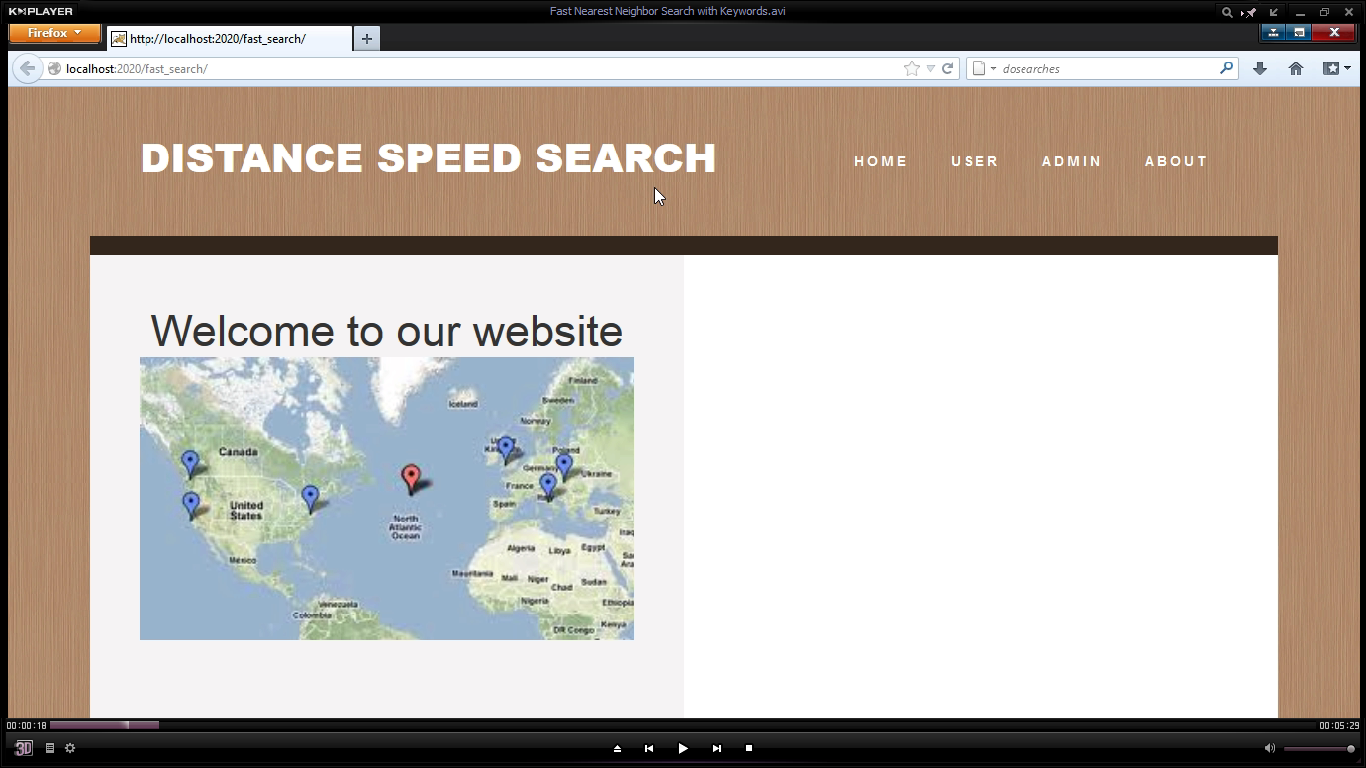
Add Location Details

**USER:**

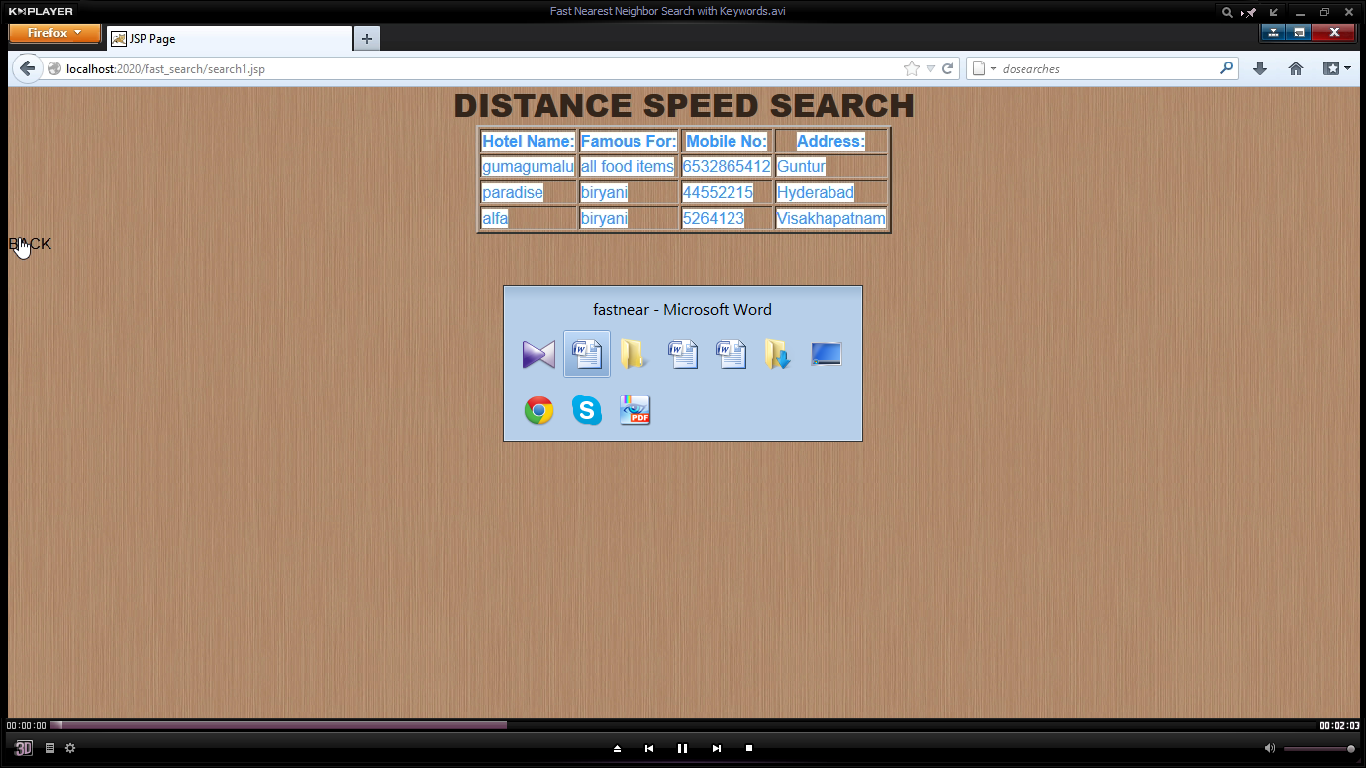
Search Keyword

**Database**

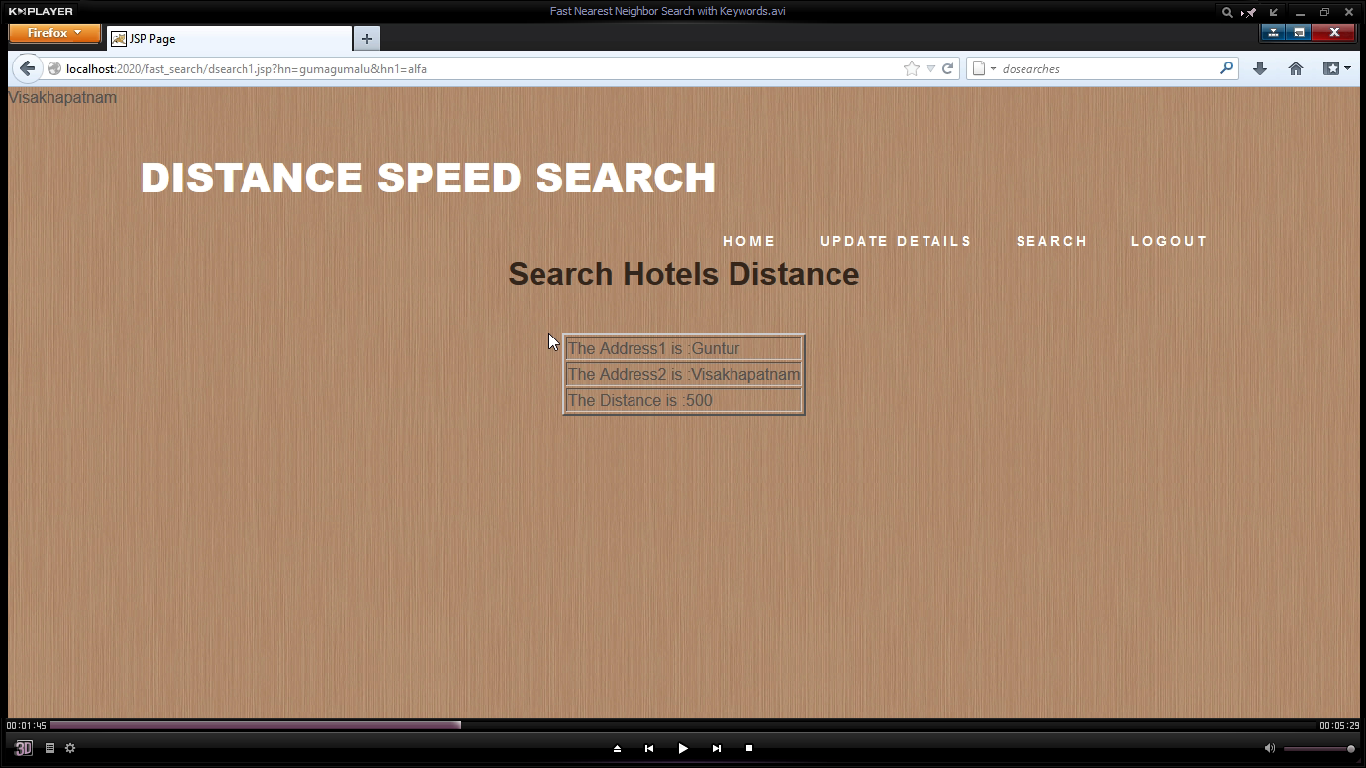
**SCREENSHOTS**

****

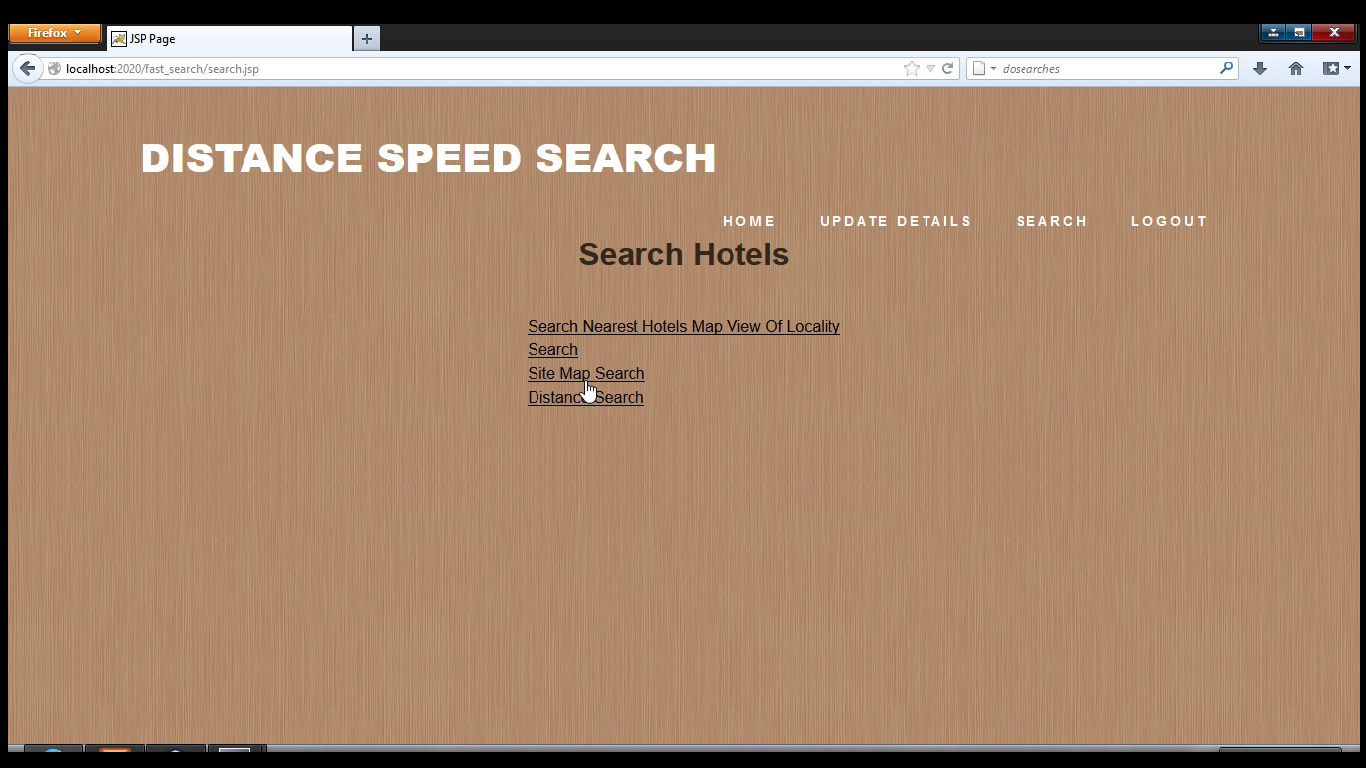
**2)**

****

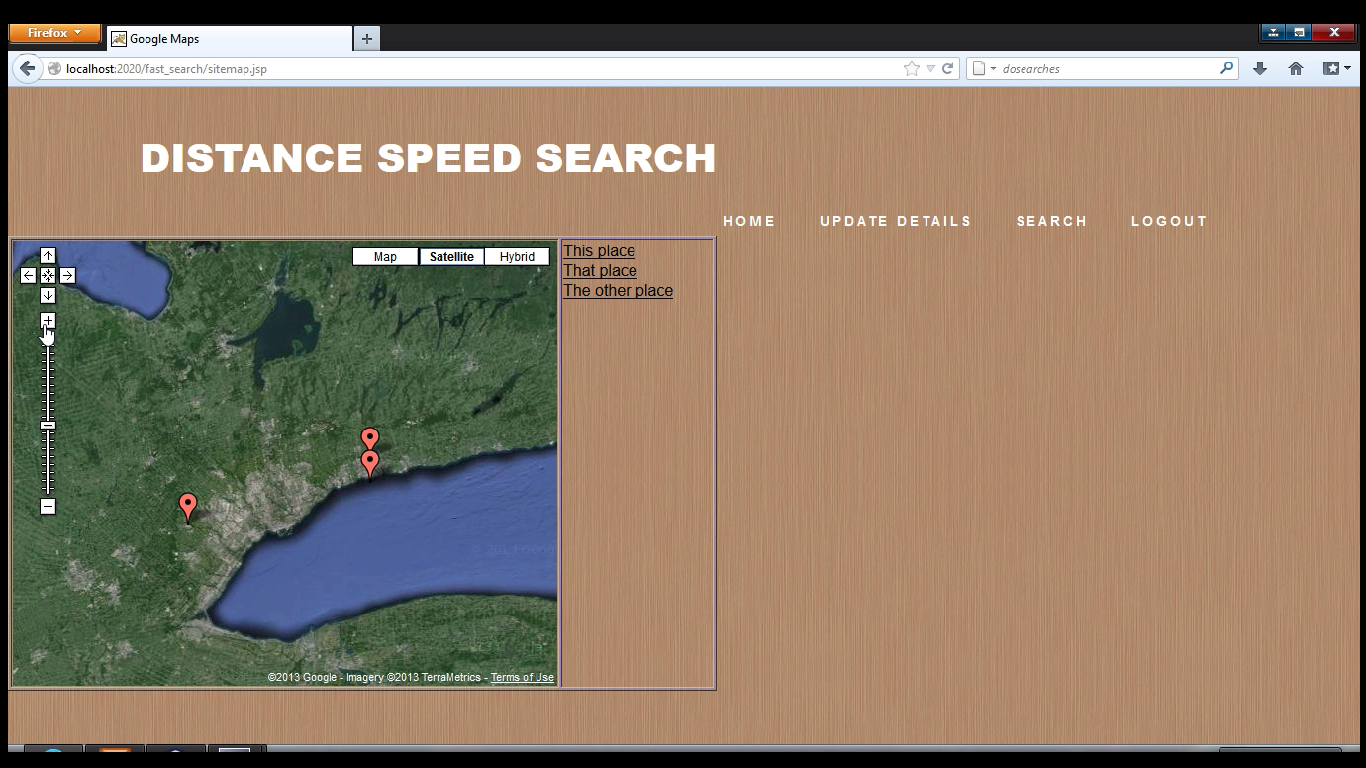
**3)**

****

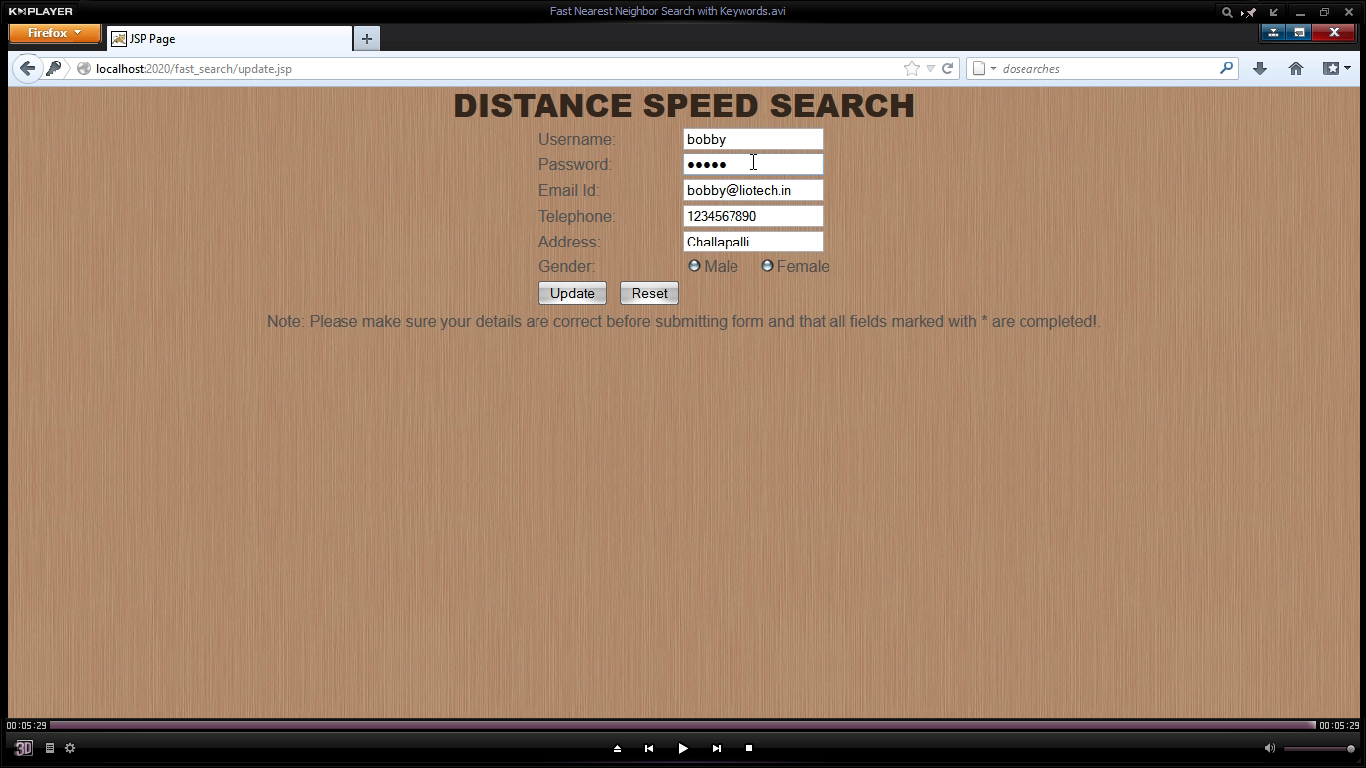
**4)**

****

**5)**

****

**6)**

****

CONCLUSION

We have seen plenty of applications calling for a search engine that is able to efﬁciently support novel forms of spatial queries that are integrated with keyword search. The existing solutions to such queries either incur prohibitive space consumption or are unable to give real time answers. In this paper, we have remedied the situation by developing an access method called the spatial inverted index (SI-index). Not only that the SI-index is fairly space economical, but also it has the ability to perform keyword-augmented nearest neighbor search in time that is at the order of dozens of milli- seconds. Furthermore, as the SI-index is based on the conventional technology of inverted index, it is readily incorporable in a commercial search engine that applies massive parallelism, implying its immediate industrial merits.