ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Novel Method for Fast Search from Large Databases

G.Swathi
M.Tech,
Dept of CSE,
Prasad Engineering College,
Jangom (PECJ).

Dr.K.Babu Rao
Professor,
Dept of CSE,
Prasad Engineering College,
Jangom (PECJ).

M.Srikanth
Assistance Professor,
Dept of CSE,
Prasad Engineering College,
Jangom (PECJ).

Abstract:

Conventional abstraction queries, like vary search and nearest neighbor retrieval, involve solely conditions on objects' geometric properties. Today, several trendy applications involve novel kinds of queries that aim to seek out objects satisfying each a abstraction predicate, and a predicate on their associated texts. as an example, rather than considering all the restaurants, a nearest neighbor question would instead elicit the eating house that's the highest among those whose menus contain "steak, spaghetti, brandy" all at an equivalent time. Presently the most effective resolution to such queries is predicated on the IR2-tree, which, as shown during this paper, features a few deficiencies that seriously impact its potency. Impelled by this, we tend to develop a replacement access methodology known as the abstraction inverted index that extends the standard inverted index to address flat knowledge, and comes with algorithms that may answer nearest neighbor queries with keywords in real time. As verified by experiments, the projected techniques outgo the IR2-tree in question latent period considerably, typically by an element of orders of magnitude.

Keywords:

R-tree, UML, diagrams, brandy, index, R-tree

I. 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 brand new 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 coordinates 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 Rtrees 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.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

II. EXISTING SYSTEM:

Spatial queries with keywords have not been extensively 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 Neighbors, 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 as described in Section 6.1. Building the non leaf 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 non leaf levels add only a small amount to the overall space overhead because, in an R-tree, the number of non leaf nodes is by far lower than that of leaf nodes 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 as described in Section Building the non leaf 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 note worthy that the non leaf levels add only a small amount to the overall space overhead because, in an R-tree, the number of non leaf nodes is by far lower than that of leaf nodes.

III. PROPOSED SYSTEM:

A spatial info manages dimensional objects (such as points, rectangles, etc.), and provides quick access to those objects supported totally different choice criteria. The importance of spatial databases is mirrored by the convenience of modeling entities of reality in an exceedingly geometric manner. for instance, locations of restaurants, hotels, hospitals so on square measure typically described as points in an exceedingly map, whereas larger extents like parks, lakes, and landscapes typically as a mix of rectangles.

several functionalities of a spatial info square measure helpful in varied ways in which in specific contexts. as an example, in an exceedingly geographics system, vary search will be deployed to search out all restaurants in an exceedingly sure space, whereas nearest neighbor retrieval will discover the eating place nearest to a given address.

Furthermore, because the SI-index relies on the traditional technology of inverted index, it's without delay incorporable in an exceedingly business computer programme that applies large similarity, implying its immediate industrial deserves.

IV. 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.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

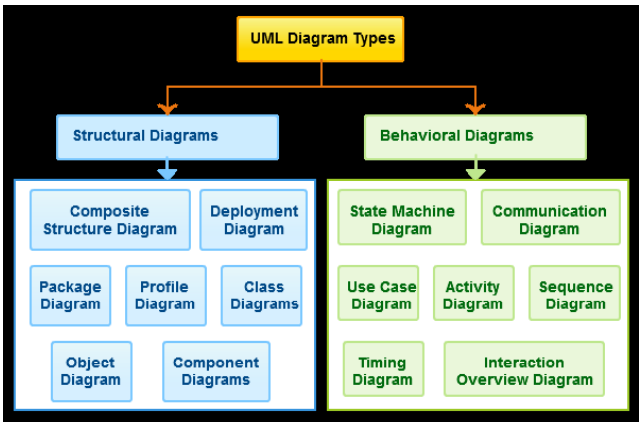


Figure.2. UML Diagram Types

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

1.Things
2.Relationships
3.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 and 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

4.1. Structural Diagrams:

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

4.2. 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

4.3. 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.

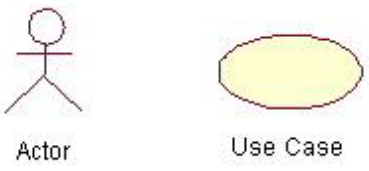


Figure.3. Use Case Diagram

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:

1. Use cases
2. Actors
3. Dependency, Generalization, and association relationships
4. System boundary

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

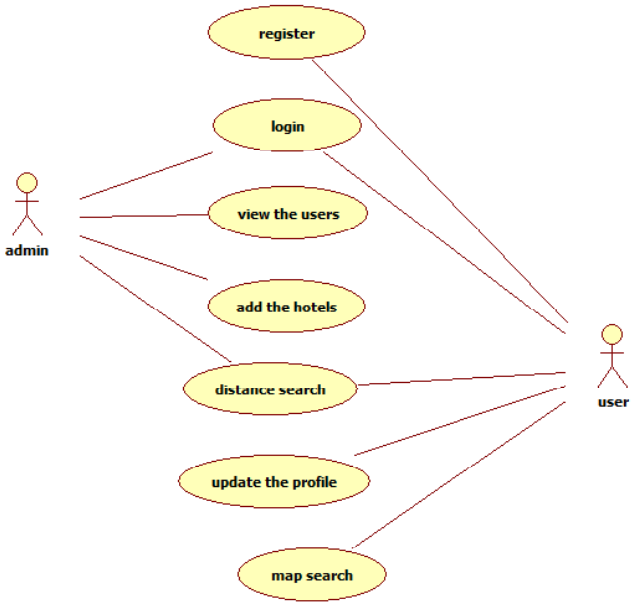


Figure.4.UML Class Diagram with Relationships

4.4. 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

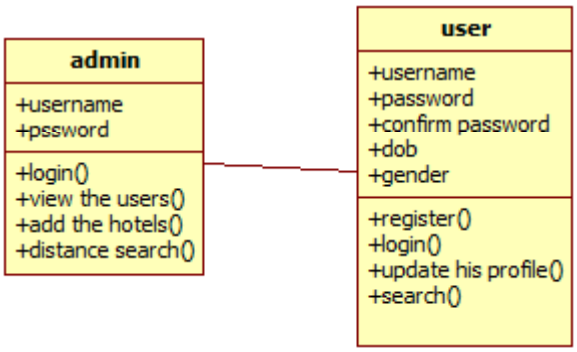


Figure.5. Class Diagram

4.5. Sequence diagram:

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

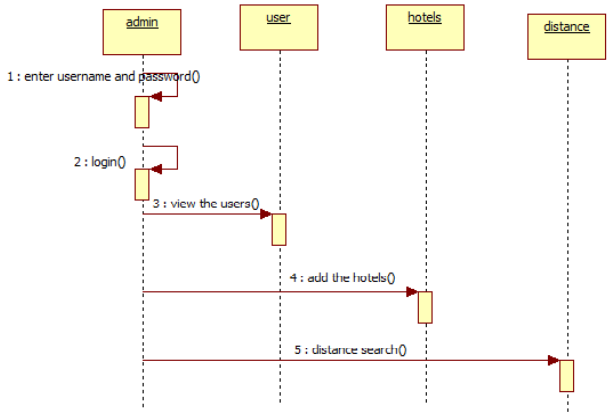


Figure.6. Basic Sequence Diagram

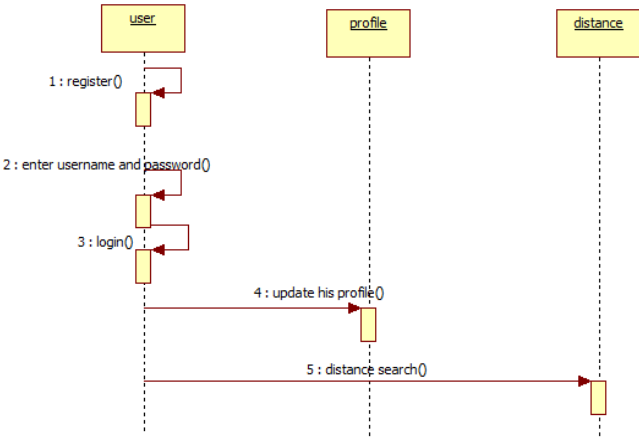


Figure.7. User Sequence Diagram

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

4.6. 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.

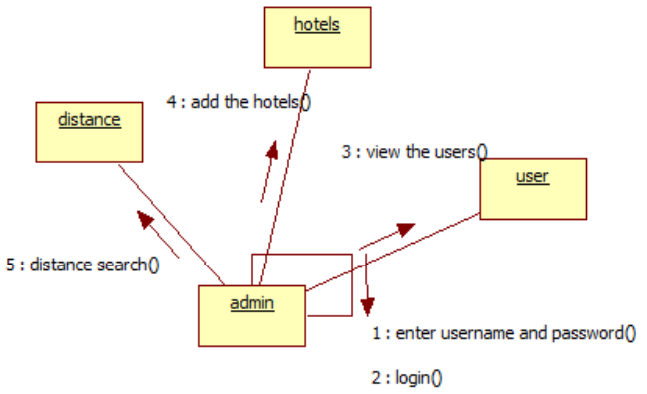


Figure.8. Collaboration Diagram

4.7. 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.

4.8. 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. 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.

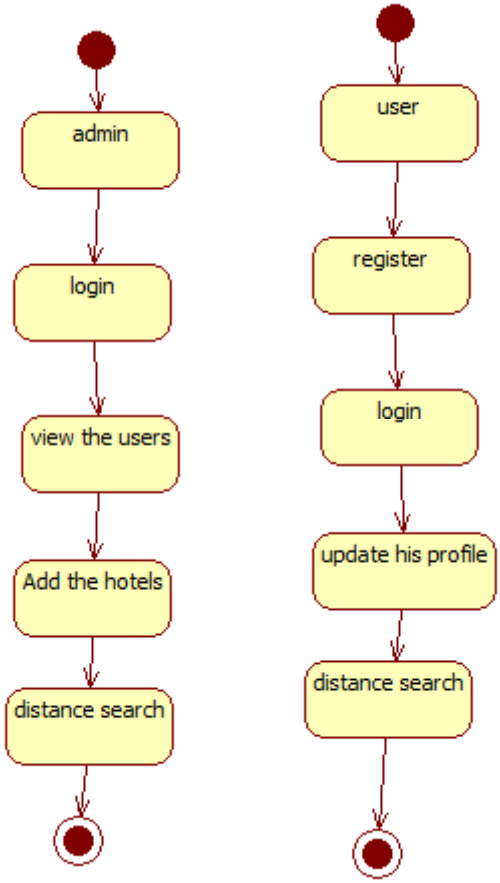


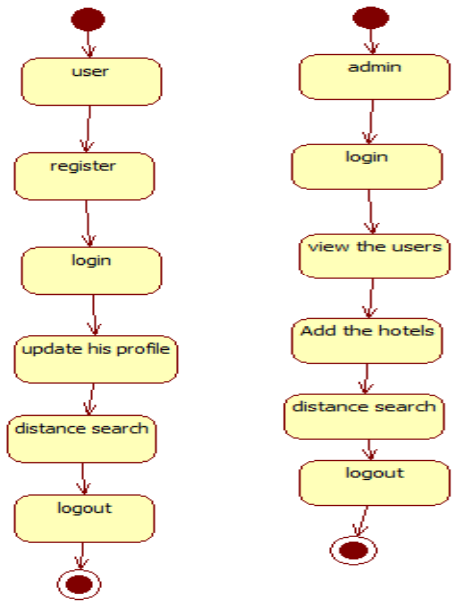
Figure.9. Admin & User State machine diagrams

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.

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.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal



Figu re.10. Admin & User Activity diagrams

4.9. Component diagram:

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

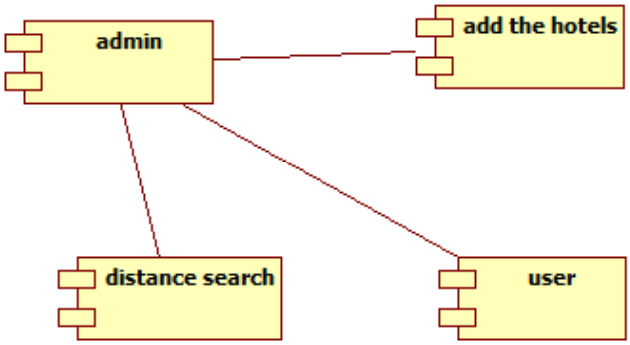


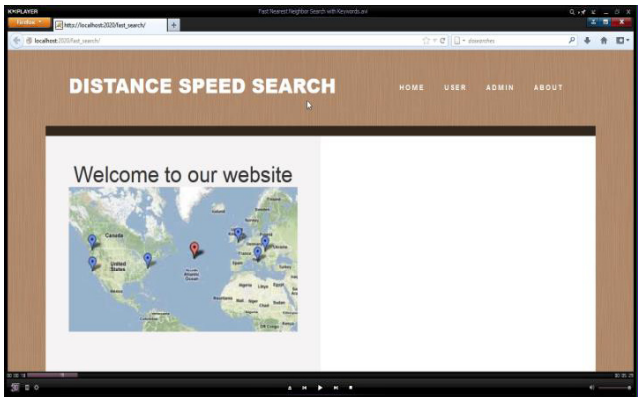
Figure.11. Component diagram

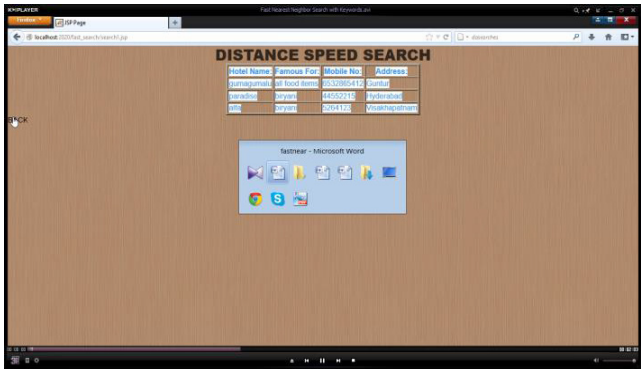
4.10. Deployment Diagram:

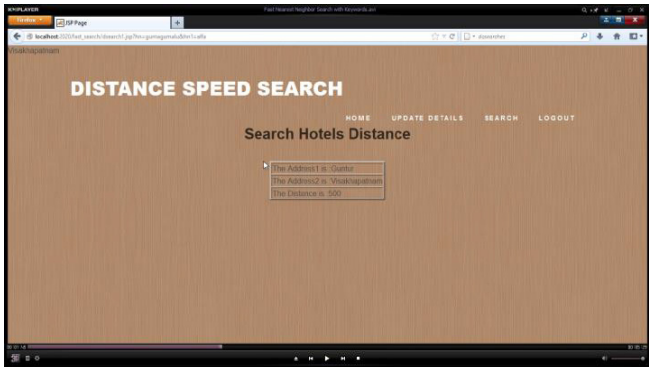
A deployment diagrams shows the hardware of your system and the software in those hardware. Deployment diagrams are useful when your software solution is deployed across multiple machines with each having a unique configuration. Below is an example deployment diagram.

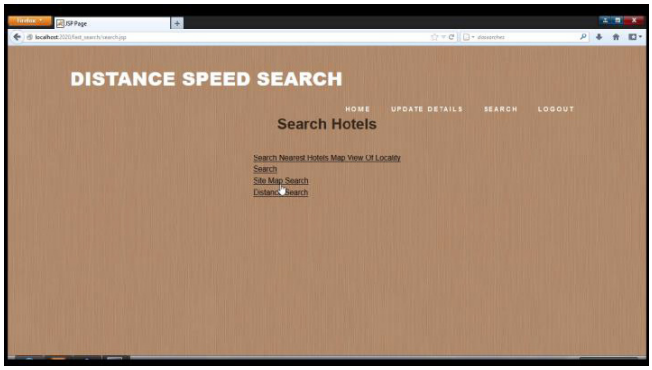
Figure.12. Deployment diagram

SCREENSHOTS

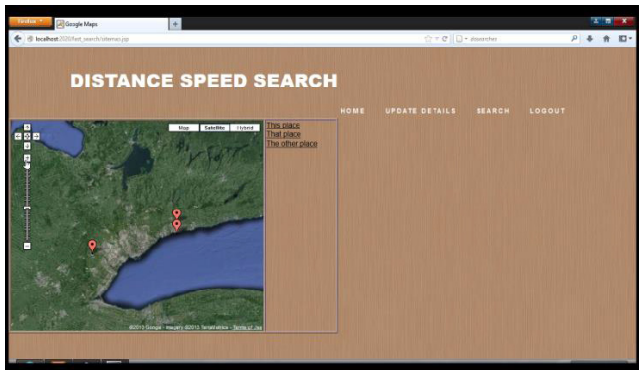


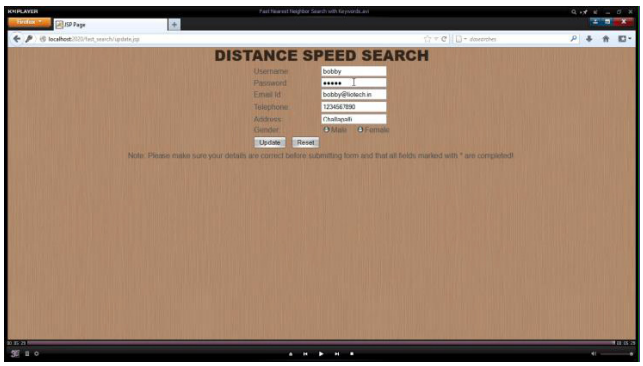






ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal





V. TESTING:

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet –undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing. The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that as a probability of finding a yet undiscovered error. A successful test is one that uncovers a yet undiscovered error. Any engineering product can be tested in one of the two ways:

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product.It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

VI. TYPES OF TESTS:

6.1. Unit testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2. Integration testing:

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.3. Functional test:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items: Valid Input : identified classes of valid input must be accepted.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Invalid Input : identified classes of invalid input must be rejected.
Functions : identified functions must be exercised.
Output : identified classes of application outputs must be exercised.

Systems/Procedures:

interfacing systems or procedures must be invoked. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing.Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

6.4. System Test:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

6.5. White Box Testing:

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

6.6. Black Box Testing:

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

6.1.1. Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach:

Field testing will be performed manually and functional tests will be written in detail.

Test objectives :

1.All field entries must work properly.
2. Pages must be activated from the identified link.
3. The entry screen, messages and responses must not be delayed.

Features to be tested :

1. Verify that the entries are of the correct format
2. No duplicate entries should be allowed
3. All links should take the user to the correct page.

Integration Testing:

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results:

All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing:

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Test Results:

All the test cases mentioned above passed successfully. No defects encountered.

Test case number	Test case	Input	Expected output	Obtained output
1	User Login	Give user name and password	User page open	user page open
2	Admin Login	Give patient mane and password	Admin page is open	Admin page is open
3	Distance search	Give the search name	Search result	Search result
4	Hotel search	Give the hotel name	Search result	Search result
5	Registration	If we wont enter any field get the message	Get the user friendly message	Get the user friendly message
6	Registration	Enter all the fields	Get the success message	Login page opened

VII. 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 milliseconds. 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.

REFERENCES :

[1] S. Agrawal, S. Chaudhuri, and G. Das. Dbxplorer: A system for keyword-based search over relational databases. In Proc. of International Conference on Data Engineering (ICDE), pages 5–16, 2002.

[2] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In Proc. of ACM Management of Data (SIGMOD), pages 322–331,1990.

[3] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudar- shan. Keyword searching and browsing in databases using banks. In Proc. of International Conference on Data ngineering (ICDE), pages 431–440, 2002.

[4] X. Cao, L. Chen, G. Cong, C. S. Jensen, Q. Qu, A. Skovsgaard, D. Wu, and M. L. Yiu. Spatial keyword querying. In ER, pages 16–29, 2012.

[5] X. Cao, G. Cong, and C. S. Jensen. Retrieving top-k prestige-based relevant spatial web objects. PVLDB, 3(1):373–384, 2010.

[6] X. Cao, G. Cong, C. S. Jensen, and B. C. Ooi. Collective spatial keyword querying. In Proc. of ACM Management of Data (SIG-MOD), pages 373–384, 2011.

[7] B. Chazelle, J. Kilian, R. Rubinfeld, and A. Tal. The bloomier filter: an efficient data structure for static support lookup tables. In Proc. of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 30–39, 2004.

[8] Y.-Y. Chen, T. Suel, and A. Markowetz. Efficient query processing in geographic web search engines. In Proc. of ACM Management of Data (SIGMOD), pages 277–288, 2006.

[9] E. Chu, A. Baid, X. Chai, A. Doan, and J. Naughton. Combining keyword search and forms for ad hoc querying of databases. In Proc. of ACM Management of Data (SIGMOD), 2009.

[10] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant s