G L Bajaj Institute of Technology and Management, Greater Noida

A
PROJECT
On
CAR-POOLING

Submitted in fulfillment of the requirement for the award of the degree of
Master of Computer Applications (MCA)

(2018-19)

Under the supervision of
Asst. Prof. Sandeep shrivastava

Submitted By

Abhishek kumar sharma
Roll No – 1719214810

DR. A. P. J. ABDUL KALAM TECHNICAL UNIVERSITY,
UTTAR PRADESH, LUCKNOW

ACKNOWLEDGEMENT

I would like to thank respected Mr. Sandeep shrivastava for giving me such a wonderful

Opportunity to expand my knowledge for my own branch and giving me guidelines to present a seminar report. It helped me a lot to realize of what we study for.

Secondly, I would like to thank my parents who patiently helped me as i went through my work and helped to modify and eliminate some of the irrelevant or un-necessary stuffs.

Thirdly, I would like to thank my friends who helped me to make my work more organized and well-stacked till the end.

Next, I would thank Microsoft for developing such a wonderful tool like MS Word. It helped my work a lot to remain error-free.

Last but clearly not the least, I would thank The Almighty for giving me strength to complete my report on time.

CERTIFICATE OF ACCEPTANCE

TABLE OF CONTENTS

Introduction

Car pooling System is a Java web project which reduces the misery of travelers and make them find cars in short period of time.

Car pooling, is an application of finding car in which drivers who are traveling to work alone can ask for fellow passengers through our application.

For those who use public-transport system to go to work daily can use this application to find drivers who are traveling to the same destination in a short path.

Carpooling is also a more environmentally friendly and sustainable way to travel as sharing journeys reduces air pollution, carbon emissions, traffic congestion on the roads, and the need for parking spaces.

Authorities often encourage carpooling, especially during periods of high pollution or high fuel prices.

Existing System:

In the existing system there is no good communication between administration and the car drivers. The older system is not user friendly. Administrator not able to view the requests of the users. Security is less compared to existing system. Details of the routes are not given online.

Proposed System:

The proposed system is user friendly. Good communication is maintained between admin and driver. All the users requests can be viewed by the administrator immediately. Details of the driver and car are maintained in the database. High level security is assigned in the proposed system.

Modules:

(1)Admin:

Admin gets login with a valid username and password. Admin can assign the services to the employees with details. Admin can view the details of the employee.

(2)Employee:

Employee should get registered with the site by giving all his personal details. Employee can get login with unique username and password. Employee can view all the services provided by the administrator.

OBJECTIVE

The main Objective of the car pooling System is to Manage the Details of Car,Booking,Payment,Destination,Share.It Manage all the information about Car,Customer,Share.

The Projects is totally Built at Administrative end and Thus only the administrative is guaranteed the access.

The purposed of the project is to build an Application Programme to reduce the manual work for managing the Car,Booking,Customer,Payment.It track all the Details about Payment,Destination,Share.

The objective of the project is to present a web based application which provides a communication platform between car owners and passengers.

Car owners will be able to post a notice announcing that (s)he has been traveling between some particular locations regularly or just once, to search

a travel-mate in order to reduce the ride costs. An example of a post described above is like that: “On 12th of July, at morning, I am driving from Ankara to Izmir. My Car is Honda Civic. 3 seats are available. Non-Smoker, male passengers are needed.

Functionalists Provided By Car Pooling:-

a) Provided the Searching Facilities Based on various Factor.Such as

b) Car,Share,Booking,Payment.

c) Car Pooling also Manage the Destination Details Online for booking Details,payment Detail,Car.

d) It track all the information Of the Customer,Destination,Booking.

e) Manage the information Of the Customer.

f) Show the information and description of the Customer.

g) Manage the information of the Car.

h) Editing,Adding and Updating of the Record is improved Which Results in proper resource Management of Car Data.

SCOPE

The objective of the project is to present a web based application which provides a communication platform between car owners and passengers.

Car owners will be able to post a notice announcing that (s)he has been traveling between some particular locations regularly or just once, to search

a travel-mate in order to reduce the ride costs. An example of a post described above is like that: “On 12th of July, at morning, I am driving from Ankara to Izmir. My Car is Honda Civic. 3 seats are available. Non-Smoker, male passengers are needed.

No pets allowed! Only one luggage. Cost is 12 Euros. ” Also, passengers will be able to search for a ride suitable to their situation. Benefits with respect to the drivers:

• The driver who will already make that trip on that day, will reduce his/her travel costs.

• For the ones that do not like traveling alone, will have the chance to find a travelmate.

• Shared driving carpooling can also reduce driving stress. Benefits with respect to the passengers:

• Passenger will have the chance to travel at lower costs than train or bus.

• They will make their trip with the comfort of an automobile.

• Avoiding lonely trips also applies for passenger. Benefits to environment and economy:

• Carpooling was encouraged to save oil. In reducing the number of cars on the road, carpooling decreases pollution and the need for parking space, and in a global perspective, reduces greenhouse gas emissions

• For the ones that do not like traveling alone, will have the chance to find a travelmate.

• Shared driving carpooling can also reduce driving stress. Benefits with respect to the passengers:

• They will make their trip with the comfort of an automobile.

• Avoiding lonely trips also applies for passenger. Benefits to environment and economy.

• Carpooling was encouraged to save oil. In reducing the number of cars on the road, carpooling decreases pollution and the need for parking space, and in a global perspective, reduces greenhouse gas emissions

SOFTWARE ENGINEERING PARADIGMS

Requirement and analysis:-

Huge changes in requirements are not allowed since it can damage database integrity and overall design.

Simple updates can be done avoiding any harm. Risk factors should be considered when adding new functionalities.

Also budget plan have to be reconsidered with respect to the additional requirements type. Project schedule should not be delayed by some determined time.

Reports should be revised by the related department.

1) **Functional requirements** :

a) General application requirements :-

Login Since all the operations that can be done using the application requires both the driver and passenger to be logged in, they can use the login forms of either Google Plus or Facebook. For this matter, the user is prompted to connect the app to his account and then proceed for sign in/up. After the user authorizes the application to access his social media account, the server retrieves his info. If he has never logged to the application before, a new account is created for him.

Modify profile information All users can modify their profile information. The profile information contain: name, phone number, email, type/color of car if any. The user can easily edit these information in order to be contacted and recognized. TERM DEFINITION DRIVER Any person that owns a car and wants to go from one place to another and publishes his trip on the application. PASSENGER Any person that doesn't own a car and wants to join a driver in a trip he posted and agrees to all the condition specified (price and general behavior). REGULAR TRIP A onetime long distance planned travel between two point (usually cities) with a defined departure time and price. FREQUENT TRIP A short to medium distance frequent (daily/weekly) between a neighborhood and a workplace, school or other point of interest. 10

Social media sharing In order to attract more users to the application and help users find passengers, users should be able to share their activity on the application on social media. A suggestion for sharing trips' creation, trips' registration or check in should pop-up whenever

those previous actions are performed. The sharing should be authorized by the users and not done automatically by the application in order not to spam the users' account and gain the users' confidence.

Rate driver/passenger Both the driver and passenger can rate each other in other to gain reputation. The importance of the rating is to encourage users to be helpful and nice during the trip so that they gain popularity in the application. It is also a way to ensure users of who can be trusted or not. The ratings represent a relative guarantee for the users to trust each other.

Regular trips

Create new regular trip The driver can create a new trip to be displayed when passengers search for trips. The application will prompt the driver or information of the regular trip which consists of destination, origin, meeting point (which can be pointed in a map), departure time/date , estimated arrival time and traveling preferences (number of free spots, price, size of bags, smoking/non-smoking, pets, stops …). After providing this information, the user publishes it in order to find passengers. Upon the creation of the trip, a user can share the trip he just created in social media to find passengers to drive with.

Search for regular trips and reservation When a passenger needs to find a driver for a destination, he can use a search form which asks for destination, origin, departure date/time. He can also specify the travelling preferences. When he finds 11

Check-in trip Whenever the driver or passenger arrive to the meeting point at the time agreed upon, he can check-in the meeting point in order to notify the other user and to show his punctuality. The application will use the devices GPS in order to make sure that the users are in the meeting point. When somebody checks in, a notification is sent to all the carpoolers saying that somebody is in the meeting point.

Frequent trips

Add frequent trip The driver can create a frequent trip where they show the origin and destination, departure and return times in addition to the frequency (daily and weekly).

Search frequent trips A passenger can search for a frequent that he can join. The passenger should specify the departing neighborhood, destination, departure times and frequency. The application will try to match it with the best trip. If the passenger is satisfied, he can register to the frequent and will be given the contact of the other members.

Use case diagram

Non-functional requirements

Performance The application has to offer a very quick response time as the meeting between the driver and passengers is done through notifications. In other words, the server should be able to treat notifications and propagate them instantly. The application should handle 1000 users sending queries at the same time.

Scalability The application should respond properly to a high increase of users. It should be able to handle from 10 000 users to 100 000 users. And also from 100 000 to one millions users.

Extensibility The application should by extensible in order to support multiple platforms including iOS, Windows Phone and Web.

Availability Since a lot of information about the trips and check in are available in the application, it has to be highly available and guarantees a good server up-time. The server should allow only 1 hour down time per year which is 99.99% up-time.

Privacy and Security The application should ensure the privacy of the users including the trips they take part in, their social media accounts and their accounts. The login system should also be robust where only authorized users can post and edit their own information.

Maintainability Since the application may be developed in the future by adding other features, it should be easily maintainable.

Analysis:

Is home-office car-pooling (ride-sharing) feasible for employees ? To answer this, we need to know the following:

1. Where are employees residing ?

2. How are they commuting today ?

3. Are the current commuting modes most efficient ? (Money, time, convenience and safety)

4. Can ride-sharing be better ?

5. Is ride-sharing possible within employees ? Are there enough passengers and car-owners in the same direction ? Study Intent

2. Data collection Total employees : 1515 Data collected for : 967 Data collected : Emp#, home location, mode of transport For this study, all references and percentages are calculated with 967 as basis. For security reasons, employee name, gender and other details were not shared. Data collected & provided by admin team Study inputs

3. 1. Evaluated of all major routes from Delhi/NCR to (Gurgaon office) 2. Evaluation of public transportation options in all these possible routes 1. 1st usage option : Metro + rickshaws 2. 2nd usage option : Buses 3. 3rd usage option : Auto’s / Taxi’s 3. Appx. travel cost calculation, per route, based on 1. Official metro fares available on Delhi Metro’s website 2. Buses fares (from usage experience) 3. Rickshaw fares (average values - as this is a not regulated by govt. practically) 4. Employee segmentation 1. Assignment of all employees to these routes 2. Current usage modes 5. Feasibility analysis for every route wrt ’s employee base 6. Cost comparison based on current usage and availability and usage of FV 7. Outputs deduced based on all the above steps Study methodology

4. 51% employees stay with Gurgaon 42% employees stay in Delhi 55% employees stay within 20kms 79% employees stay within 30 kms 1. Where are employees residing ?

5. Further division of 2 main cities 1. Where are employees residing ?

6. 80% employees use personal or public transportation options. 20% employees (only) use office provided transportation 2. How are they commuting today ?

7. 70% of employees are residing in locations NOT suited to use the Metro These employees are either staying in areas with NO metro connectivity (51% in GGN) or , staying in areas which have a huge detour if metro is used (17.37%) Office is located appx. 2.5 kms from nearest metro/bus station Most employees would use a Rickshaw to reach the office after reaching the nearest metro/bus station. 3. Are the current commuting modes most efficient ?

8. 4. Can ride sharing be better ? With ride sharing employees 1. Travel point to point (directly home to office) - do not need multiple transportation mechanisms - save time - save money (in some cases) - becomes more convenient 2. Travel safe - employees travel with each other 3. Increase inter/intra team coordination 4. Reduce load on public infrastructure and emissions 5. Increase productivity

9. 5. Is ride-sharing possible within employees ? 68.56% of employees come from the same 10 directions In ALL these 10 directions, ride sharing is - cheaper - safer - time efficient - more convenient The rest 30% of employees will have to pay a little more to ride-share

10. Ride-sharing will be more COST EFFECTIVE for over 70% employees from DAY 1 Added employee benefits - Employee safety - High convenience With time, the remaining 30% will understand the non- financial benefits and also join in.

FEASIBILITY STUDY:

The system must be evaluated from the technical point of view first. The assessment of this feasibility must be based on an outline design of the system requirement in the terms of input, output, programs and procedures. Having identified an outline system, the investigation must go on to suggest the type of equipment, required method developing the system, of running the system once it has been designed.

Technical issues raised during the investigation are:

- Does the existing technology sufficient for the suggested one?
- Can the system expand if developed?

The project should be developed such that the necessary functions and performance are achieved within the constraints. The project is developed within latest technology. Through the technology may become obsolete after some period of time, due to the fact that never version of same software supports older versions, the system may still be used. So there are minimal constraints involved with this project. The system has been developed using Java the project is technically feasible for development.

DESIGN:

Design is the first step into the development phase for any engineered product or system. Design is a creative process. A good design is the key to effective system. The term "design" is defined as "the process of applying various techniques and principles for the purpose of defining a process or a system in sufficient detail to permit its physical realization". It may be defined as a process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permit its physical realization. Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm that is used. The system design develops the architectural detail required to build a system or product. As in the case of any systematic approach, this software too has undergone the best possible design phase fine tuning all efficiency, performance and accuracy levels. The design phase is a transition from a user oriented document to a document to the programmers or database personnel. System design goes through two phases of development: Logical and Physical Design.

4.1 LOGICAL DESIGN:

The logical flow of a system and define the boundaries of a system. It includes the following steps:

- Reviews the current physical system – its data flows, file content, volumes, Frequencies etc.
- Prepares output specifications – that is, determines the format, content and Frequency of reports.
- Prepares input specifications – format, content and most of the input functions.

- Prepares edit, security and control specifications.
- Specifies the implementation plan.
- Prepares a logical design walk through of the information flow, output, input, Controls and implementation plan.
- Reviews benefits, costs, target dates and system constraints.

4.2 PHYSICAL DESIGN:

Physical system produces the working systems by define the design specifications that tell the programmers exactly what the candidate system must do. It includes the following steps.

- Design the physical system.
- Specify input and output media.
- Design the database and specify backup procedures.
- Design physical information flow through the system and a physical design
- Plan system implementation.
- Prepare a conversion schedule and target date.
- Determine training procedures, courses and timetable.
- Devise a test and implementation plan and specify any new hardware/software.
- Update benefits , costs , conversion date and system constraints

Design/Specification activities:

- Concept formulation.
- Problem understanding.
- High level requirements proposals.
- Feasibility study.
- Requirements engineering.
- Architectural design.

4.3 MODULE DESIGN

- ✓ Search Ads
- ✓ Poster
- ✓ Registration
- ✓ Administrator
- ✓ Login

4.4 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it

provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

- Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
- It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
- When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

4.5 OUTPUT DESIGN

- A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient

and intelligent output design improves the system's relationship to help user decision-making.

- Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
- Select methods for presenting information.
- Create document, report, or other formats that contain information produced by the system.

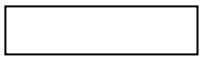
The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

4.6 Data Flow Diagram

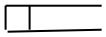
The Data flow Diagram shows the flow of data. It is generally made of symbols given below:

(1) A **square** shows the Entity: -



(2) A **Circle** shows the Process: -

(3) An **open Ended Rectangle** shows the data store: --



(4) An **arrow** shows the data flow:-

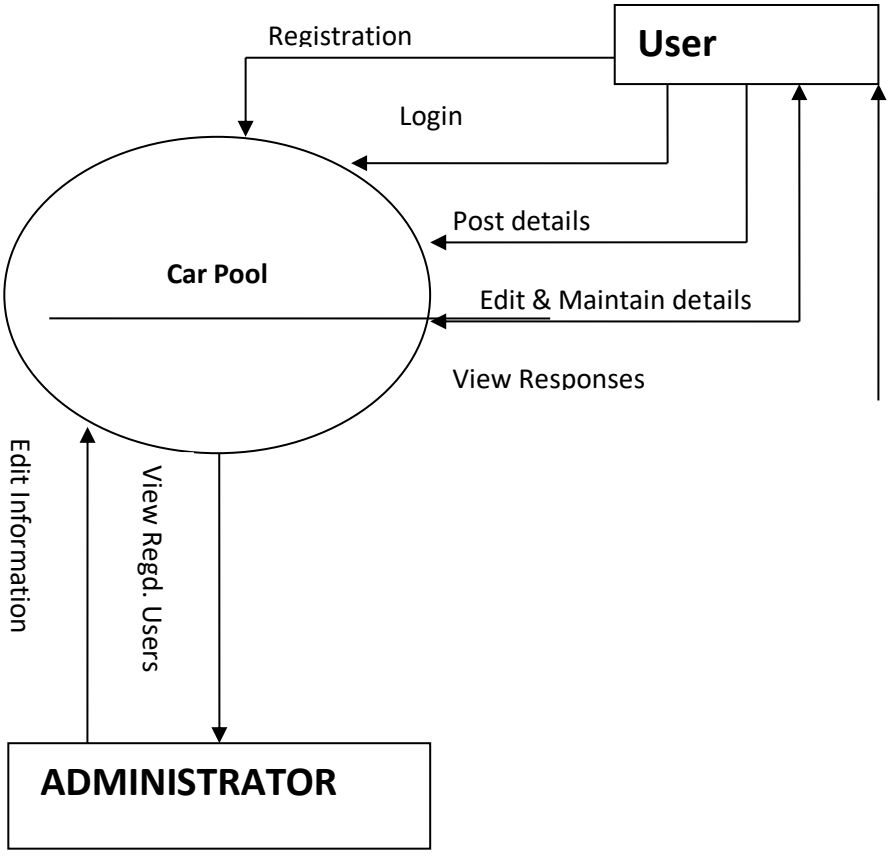
The DFD can be up to several levels.

The 0 level DFD states the flow of data in the system as seen from the outward in each module.

The first level DFD show more detail, about the single process of the 0 level DFD

The second level DFD can show even more details and so on.

Context Level DFD



4.7 DATABASE DESIGN

A database is an organized mechanism that has the capability of storing information through which a user can retrieve stored information in an effective and efficient manner. The data is the purpose of any database and must be protected.

The database design is a two level process. In the first step, user requirements are gathered together and a database is designed which will meet these requirements as clearly as possible. This step is called Information Level Design and it is taken independent of any individual DBMS.

In the second step, this Information level design is transferred into a design for the specific DBMS that will be used to implement the system in question. This step is called Physical Level Design, concerned with the characteristics of the specific DBMS that will be used. A database design runs parallel with the system design.

The organization of the data in the database is aimed to achieve the following two major objectives.

- Data Integrity
- Data independence

Normalization is the process of decomposing the attributes in an application, which results in a set of tables with very simple structure. The purpose of normalization is to make tables as simple as possible.

Normalization is carried out in this system for the following reasons.

- To structure the data so that there is no repetition of data , this helps in saving.

- To permit simple retrieval of data in response to query and report request.
- To simplify the maintenance of the data through updates, insertions,
- Deletions.
- To reduce the need to restructure or reorganize data which new application
- Requirements arise.

ROLE IN THE PROJECTS:

primary role in the project, and is responsible for its successful completion. The Developer is to ensure that the project proceeds within the specified time frame and under the established budget, while achieving its objectives. Project managers make sure that projects are given sufficient resources, while managing relationships with contributors and stakeholders.

Developer duties:

- Develop a project plan
- Manage deliverables according to the plan
- Recruit project staff
- Lead and manage the project team
- Determine the methodology used on the project
- Establish a project schedule and determine each phase
- Assign tasks to project team members
- Provide regular updates to upper management

Project Team Member

Project team members are the individuals who actively work on one or more phases of the project. They may be in-house staff or external consultants, working on the project on a full-time or part-time basis. Team member roles can vary according to each project.

Project team member duties may include:

- Contributing to overall project objectives
- Completing individual deliverables
- Providing expertise
- Working with users to establish and meet business needs
- Documenting the process

Project Sponsor

The project sponsor is the driver and in-house champion of the project. They are typically members of senior management – those with a stake in the project's outcome. Project sponsors work closely with the project manager. They legitimize the project's objectives and participate in high-level project planning. In addition, they

often help resolve conflicts and remove obstacles that occur throughout the project, and they sign off on approvals needed to advance each phase.

Project sponsor duties:

- Make key business decisions for the project
- Approve the project budget
- Ensure availability of resources
- Communicate the project's goals througout the organization

WORKING METHODOLOGY

SOFTWARE ENVIRONMENT:

JAVA

Java is a small, simple, safe, object oriented, interpreted or dynamically optimized, byte coded, architectural, garbage collected, multithreaded programming language with a strongly typed exception-handling for writing distributed and dynamicaly extensible programs.

Java is an object oriented programming language. Java is a high-level, third generation language like C, FORTRAN, Small talk, Pearl and many others. You can use

java to write computer applications that crunch numbers, process words, play games, store data or do any of the thousands of other things computer software can do.

Special programs called applets that can be downloaded from the internet and played safely within a web browser. Java a supports this application and the follow features make it one of the best programming languages.

- It is simple and object oriented
- It helps to create user friendly interfaces.
- It is very dynamic.
- It supports multithreading.
- It is platform independent
- It is highly secure and robust.
- It supports internet programming

Java is a programming language originally developed by Sun Microsystems and released in 1995 as a core component of Sun's Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java

applications are typically compiled to byte code which can run on any Java virtual machine (JVM) regardless of computer architecture.

The original and reference implementation Java compilers, virtual machines, and class libraries were developed by Sun from 1995. As of May 2007, in compliance with the specifications of the Java Community Process, Sun made available most of their Java technologies as free software under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java and GNU Class path.

The Java language was created by James Gosling in June 1991 for use in a set top box project. The language was initially called *Oak*, after an oak tree that stood outside Gosling's office - and also went by the name *Green* - and ended up later being renamed to *Java*, from a list of random words. Gosling's goals were to implement a virtual machine and a language that had a familiar C/C++ style of notation.

Primary goals

There were five primary goals in the creation of the Java language:

1. It should use the object-oriented programming methodology.
2. It should allow the same program to be executed on multiple operating systems.
3. It should contain built-in support for using computer networks.
4. It should be designed to execute code from remote sources securely.
5. It should be easy to use by selecting what were considered the good parts of other object-oriented languages.

The Java platform is the name for a bundle of related programs, or platform, from Sun which allow for developing and running programs written in the Java programming

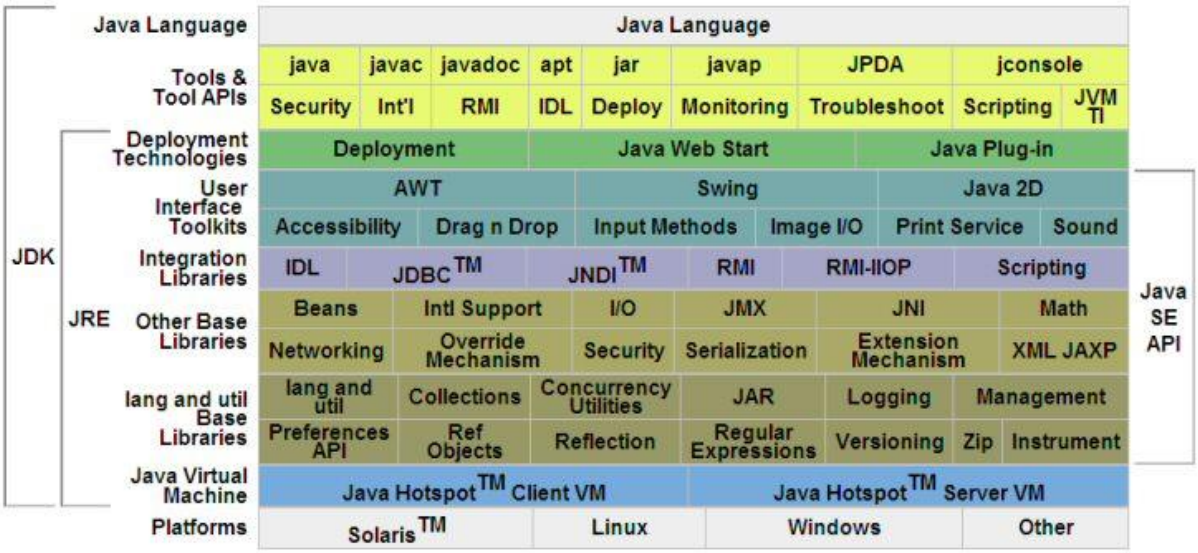
language. The platform is not specific to any one processor or operating system, but rather an execution engine (called a virtual machine) and a compiler with a set of standard libraries which are implemented for various hardware and operating systems so that Java programs can run identically on all of them.

Different "editions" of the platform are available, including:

- Java ME (Micro Edition): Specifies several different sets of libraries (known as profiles) for devices which are sufficiently limited that supplying the full set of Java libraries would take up unacceptably large amounts of storage.
- Java SE (Standard Edition): For general purpose use on desktop PCs, servers and similar devices.
- Java EE (Enterprise Edition): Java SE plus various APIs useful for multi-tier client-server enterprise applications.

The Java Platform consists of several programs, each of which provides a distinct portion of its overall capabilities. For example, the Java compiler, which converts Java source code into Java bytecode (an intermediate language for the Java Virtual Machine (JVM)), is provided as part of the Java Development Kit (JDK). The sophisticated Java Runtime Environment (JRE), complementing the JVM with a just-in-time (JIT) compiler, converts intermediate bytecode into native machine code on the fly. Also supplied are extensive libraries (pre-compiled into Java bytecode) containing reusable code, as well as numerous ways for Java applications to be deployed, including being embedded in a web page as an applet.There are several other components, some available only in certain editions.

The essential components in the platform are the Java language compiler, the libraries, and the runtime environment in which Java intermediate byte code "executes" according to the rules laid out in the virtual machine specification.



JAVA VIRTUAL MACHINE

The heart of the Java Platform is the concept of a "virtual machine" that executes Java byte code programs. This byte code is the same no matter what hardware or operating system the program is running under. There is a JIT compiler within the *Java Virtual Machine*, or JVM. The JIT compiler translates the Java byte code into native processor instructions at run-time and caches the native code in memory during execution.

The use of byte code as an intermediate language permits Java programs to run on any platform that has a virtual machine available. The use of a JIT compiler means that Java applications, after a short delay during loading and once they have "warmed up" by being all or mostly JIT-compiled, tend to run about as fast as native programs. Since JRE version 1.2, Sun's JVM implementation has included a just-in-time compiler instead of an interpreter.

Although Java programs are Platform Independent, the code of the Java Virtual Machine (JVM) that execute these programs are not. Every Operating System has its own JVM.

CLASS LIBRARIES

In most modern operating systems, a large body of reusable code is provided to simplify the programmer's job. This code is typically provided as a set of dynamically loadable libraries that applications can call at runtime. Because the Java Platform is not dependent on any specific operating system, applications cannot rely on any of the existing libraries. Instead, the Java Platform provides a comprehensive set of standard class libraries, containing much of the same reusable functions commonly found in modern operating systems.

The Java class libraries serve three purposes within the Java Platform. Like other standard code libraries, they provide the programmer a well-known set of functions to perform common tasks, such as maintaining lists of items or performing complex string parsing. In addition, the class libraries provide an abstract interface to tasks that would normally depend heavily on the hardware and operating system. Tasks such as network access and file access are often heavily dependent on the native capabilities of the platform. The Java java.net and java.io libraries implement the required native code internally, then provide a standard interface for the Java applications to perform those tasks. Finally, when some underlying platform does not support all of the features a Java application expects, the class libraries can either emulate those features using whatever is available, or at least provide a consistent way to check for the presence of a specific feature.

PLATFORM INDEPENDENCE

One characteristic, platform independence, means that programs written in the Java language must run similarly on any supported hardware/operating-system platform. One should be able to write a program once, compile it once, and run it anywhere.

This is achieved by most Java compilers by compiling the Java language code *halfway* (to Java bytecode) – simplified machine instructions specific to the Java platform. The code is then run on a virtual machine (VM), a program written in native code on the host hardware that interprets and executes generic Java bytecode. (In some JVM versions, bytecode can also be compiled to native code, either before or during program execution,

resulting in faster execution.) Further, standardized libraries are provided to allow access to features of the host machines (such as graphics, threading and networking) in unified ways. Note that, although there is an explicit compiling stage, at some point, the Java bytecode is interpreted or converted to native machine code by the JIT compiler.

The first implementations of the language used an interpreted virtual machine to achieve portability. These implementations produced programs that ran more slowly than programs compiled to native executables, for instance written in C or C++, so the language suffered a reputation for poor performance. More recent JVM implementations produce programs that run significantly faster than before, using multiple techniques.

One technique, known as *just-in-time compilation* (JIT), translates the Java bytecode into native code at the time that the program is run, which results in a program that executes faster than interpreted code but also incurs compilation overhead during execution. More sophisticated VMs use *dynamic recompilation*, in which the VM can analyze the behavior of the running program and selectively recompile and optimize critical parts of the program. Dynamic recompilation can achieve optimizations superior to static compilation because the dynamic compiler can base optimizations on knowledge about the runtime environment and the set of loaded classes, and can identify the *hot spots* (parts of the program, often inner loops, that take up the most execution time). JIT compilation and dynamic recompilation allow Java programs to take advantage of the speed of native code without losing portability.

Another technique, commonly known as *static compilation*, is to compile directly into native code like a more traditional compiler. Static Java compilers, such as GCJ, translate the Java language code to native object code, removing the intermediate bytecode stage. This achieves good performance compared to interpretation, but at the expense of portability; the output of these compilers can only be run on a single architecture. Some see avoiding the VM in this manner as defeating the point of developing in Java; however it can be useful to provide both a generic bytecode version, as well as an optimised native code version of an application.

AUTOMATIC MEMORY MANAGEMENT

One of the ideas behind Java's automatic memory management model is that programmers be spared the burden of having to perform manual memory management. In some languages the programmer allocates memory for the creation of objects stored on the heap and the responsibility of later deallocating that memory also resides with the programmer. If the programmer forgets to deallocate memory or writes code that fails to do so, a memory leak occurs and the program can consume an arbitrarily large amount of memory. Additionally, if the program attempts to deallocate the region of memory more than once, the result is undefined and the program may become unstable and may crash. Finally, in non garbage collected environments, there is a certain degree of overhead and complexity of user-code to track and finalize allocations. Often developers may box themselves into certain designs to provide reasonable assurances that memory leaks will not occur.

In Java, this potential problem is avoided by automatic garbage collection. The programmer determines when objects are created, and the Java runtime is responsible for managing the object's lifecycle. The program or other objects can reference an object by holding a reference to it (which, from a low-level point of view, is its address on the heap). When no references to an object remain, the Java garbage collector automatically deletes the unreachable object, freeing memory and preventing a memory leak. Memory leaks may still occur if a programmer's code holds a reference to an object that is no longer needed—in other words, they can still occur but at higher conceptual levels.

The use of garbage collection in a language can also affect programming paradigms. If, for example, the developer assumes that the cost of memory allocation/recollection is low, they may choose to more freely construct objects instead of pre-initializing, holding and reusing them. With the small cost of potential performance penalties (inner-loop construction of large/complex objects), this facilitates thread-isolation(no need to synchronize as different threads work on different object instances) and data-hiding. The use of transient immutable value-objects minimizes side-effect programming.

Comparing Java and C++, it is possible in C++ to implement similar functionality (for example, a memory management model for specific classes can be designed in C++ to improve speed and lower memory fragmentation considerably), with the possible cost of adding comparable runtime overhead to that of Java's garbage collector, and of added development time and application complexity if one favors manual implementation over using an existing third-party library. In Java, garbage collection is built-in and virtually invisible to the developer. That is, developers may have no notion of when garbage collection will take place as it may not necessarily correlate with any actions being explicitly performed by the code they write. Depending on intended application, this can be beneficial or disadvantageous: the programmer is freed from performing low-level tasks, but at the same time loses the option of writing lower level code. Additionally, the garbage collection capability demands some attention to tuning the JVM, as large heaps will cause apparently random stalls in performance.

Java does not support pointer arithmetic as is supported in, for example, C++. This is because the garbage collector may relocate referenced objects, invalidating such pointers. Another reason that Java forbids this is that type safety and security can no longer be guaranteed if arbitrary manipulation of pointers is allowed.

PERFORMANCE

Java's performance has improved substantially since the early versions, and performance of JIT compilers relative to native compilers has in some tests been shown to be quite similar. The performance of the compilers does not necessarily indicate the performance of the compiled code; only careful testing can reveal the true performance issues in any system.

JAVA RUNTIME ENVIRONMENT

The Java Runtime Environment, or *JRE*, is the software required to run any application deployed on the Java Platform. End-users commonly use a JRE in software packages and Web browser plugins. Sun also distributes a superset of the JRE called the Java 2 SDK (more commonly known as the JDK), which includes development tools such as the Java compiler, Javadoc, Jar and debugger.

One of the unique advantages of the concept of a runtime engine is that errors (exceptions) should not 'crash' the system. Moreover, in runtime engine environments such as Java there exist tools that attach to the runtime engine and every time that an exception of interest occurs they record debugging information that existed in memory at the time the exception was thrown (stack and heap values). These Automated Exception Handling tools provide 'root-cause' information for exceptions in Java programs that run in production, testing or development environments.

- **Java 2 Enterprise Edition (J2EE)**

The J2EE platform uses a multitier distributed application model. Application logic is divided into components according to function, and the various application components that make up a J2EE application are installed on different machines depending on the tier in the multitier J2EE environment to which the application component belongs. Figure 1-1 shows two multitier J2EE applications divided into the tiers described in the following list. The J2EE application parts shown in Figure 1-1 are presented in J2EE Components.

- Client-tier components run on the client machine.
- Web-tier components run on the J2EE server.
- Business-tier components run on the J2EE server.
- Enterprise information system (EIS)-tier software runs on the EIS server.

Although a J2EE application can consist of the three or four tiers shown in Figure 1-1, J2EE multitier applications are generally considered to be three-tiered applications because they are distributed over three different locations: client machines, the J2EE server machine, and the database or legacy machines at the back end. Three-tiered applications that run in this way extend the standard two-tiered client and server model by placing a multithreaded application server between the client application and back-end storage.

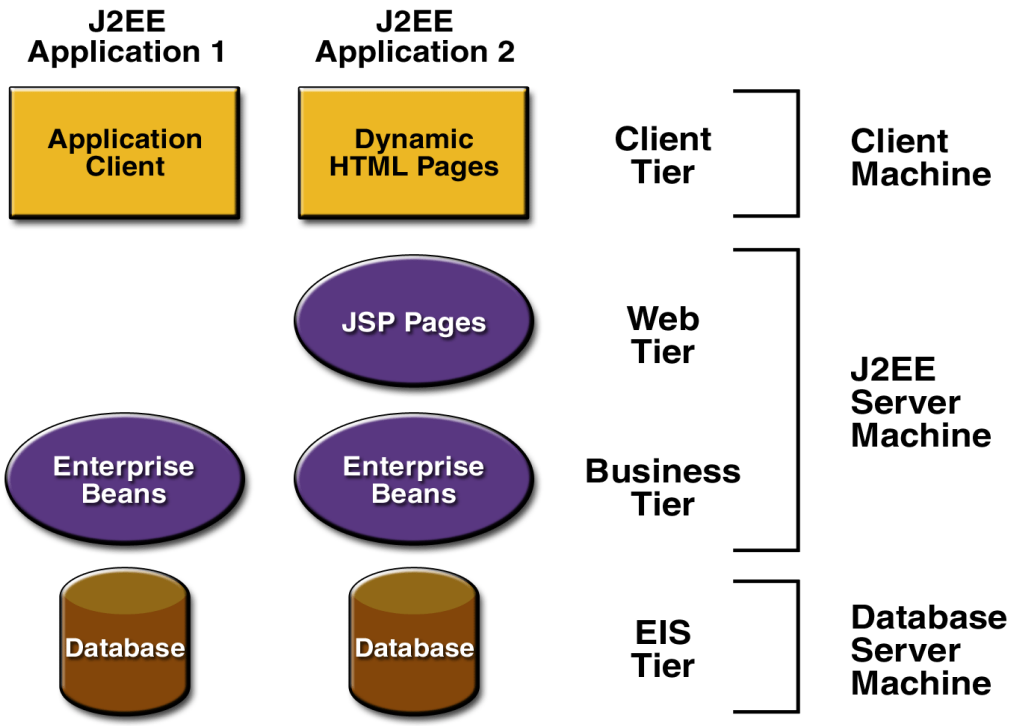


Figure 1-1 Multitier Applications

J2EE COMPONENTS

J2EE applications are made up of components. A *J2EE component* is a self-contained functional software unit that is assembled into a J2EE application with its related classes and files and that communicates with other components. The J2EE specification defines the following J2EE components:

- Application clients and applets are components that run on the client.
- Java Server and Java Server Pages (JSP) technology components are Web components that run on the server.
- Enterprise JavaBeans (EJB) components (enterprise beans) are business components that run on the server.

J2EE components are written in the Java programming language and are compiled in the same way as any program in the language. The difference between J2EE components and "standard" Java classes is that J2EE components are assembled into a J2EE application, verified to be well formed and in compliance with the J2EE specification, and deployed to production, where they are run and managed by the J2EE server.

J2EE CLIENTS

A J2EE client can be a Web client or an application client.

WEB CLIENTS

A Web client consists of two parts: dynamic Web pages containing various types of markup language (HTML, XML, and so on), which are generated by Web components running in the Web tier, and a Web browser, which renders the pages received from the server.

A Web client is sometimes called a *thin client*. Thin clients usually do not do things like query databases, execute complex business rules, or connect to legacy applications. When you use a thin client, heavyweight operations like these are off-loaded to enterprise beans executing on the J2EE server where they can leverage the security, speed, services, and reliability of J2EE server-side technologies.

APPLETS

A Web page received from the Web tier can include an embedded applet. An applet is a small client application written in the Java programming language that executes in the Java virtual machine installed in the Web browser. However, client systems will likely need the Java Plug-in and possibly a security policy file in order for the applet to successfully execute in the Web browser.

Web components are the preferred API for creating a Web client program because no plug-ins or security policy files are needed on the client systems. Also, Web components enable cleaner and more modular application design because they provide a way to separate applications

programming from Web page design. Personnel involved in Web page design thus do not need to understand Java programming language syntax to do their jobs.

APPLICATION CLIENT

A J2EE application client runs on a client machine and provides a way for users to handle tasks that require a richer user interface than can be provided by a markup language. It typically has a graphical user interface (GUI) created from Swing or Abstract Window Toolkit (AWT) APIs, but a command-line interface is certainly possible.

Application clients directly access enterprise beans running in the business tier. However, if application requirements warrant it, a J2EE application client can open an HTTP connection to establish communication with a servlet running in the Web tier.

JAVABEANS COMPONENT ARCHITECTURE

The server and client tiers might also include components based on the JavaBeans component architecture (JavaBeans component) to manage the data flow between an application client or applet and components running on the J2EE server or between server components and a database. JavaBeans components are not considered J2EE components by the J2EE specification.

JavaBeans components have instance variables and get and set methods for accessing the data in the instance variables. JavaBeans components used in this way are typically simple in design and implementation, but should conform to the naming and design conventions outlined in the JavaBeans component architecture.

J2EE SERVER COMMUNICATION

The client communicates with the business tier running on the J2EE server either directly or, as in the case of a client running in a browser, by going through JSP pages or servlets running in the Web tier.J2EE application uses a thin browser-based client or thick application client. In deciding which one to use, you should be aware of the trade-offs between keeping functionality on the client

and close to the user (thick client) and off-loading as much functionality as possible to the server (thin client). The more functionality you off-load to the server, the easier it is to distribute, deploy, and manage the application; however, keeping more functionality on the client can make for a better perceived user experience.

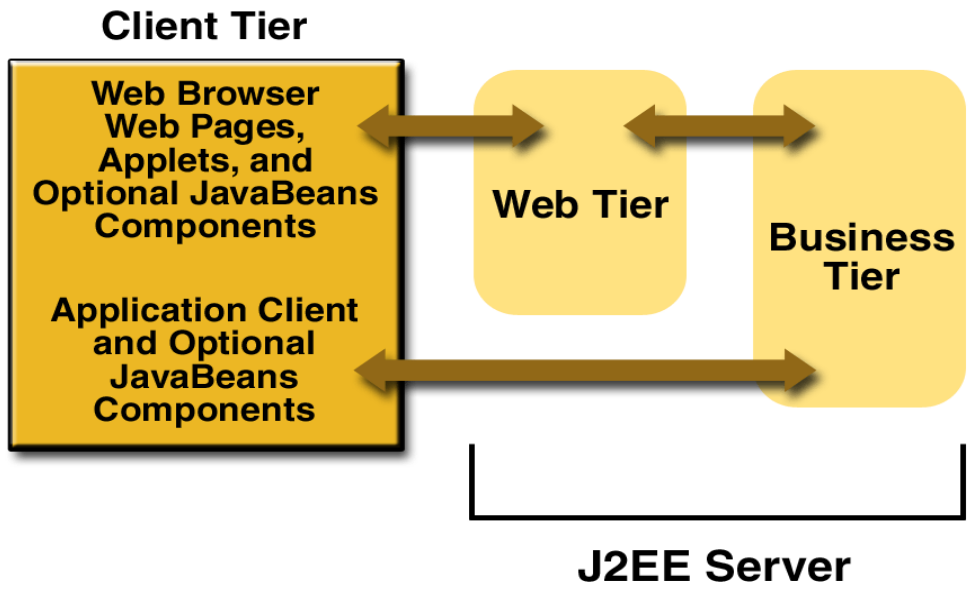


Figure 1-2 Server Communications

WEB COMPONENT

J2EE Web components can be either servlets or JSP pages. *Servlets* are Java programming language classes that dynamically process requests and construct responses. *JSP pages* are text-based documents that execute as servlets but allow a more natural approach to creating static content.Static HTML pages and applets are bundled with Web components during application assembly, but are not considered Web components by the J2EE specification. Server-side utility classes can also be bundled with Web components and, like HTML pages, are not considered Web components.Like the client tier and as shown in Figure 1-3, the Web tier might include a JavaBeans component to manage the user input and send that input to enterprise beans running in the business tier for processing.

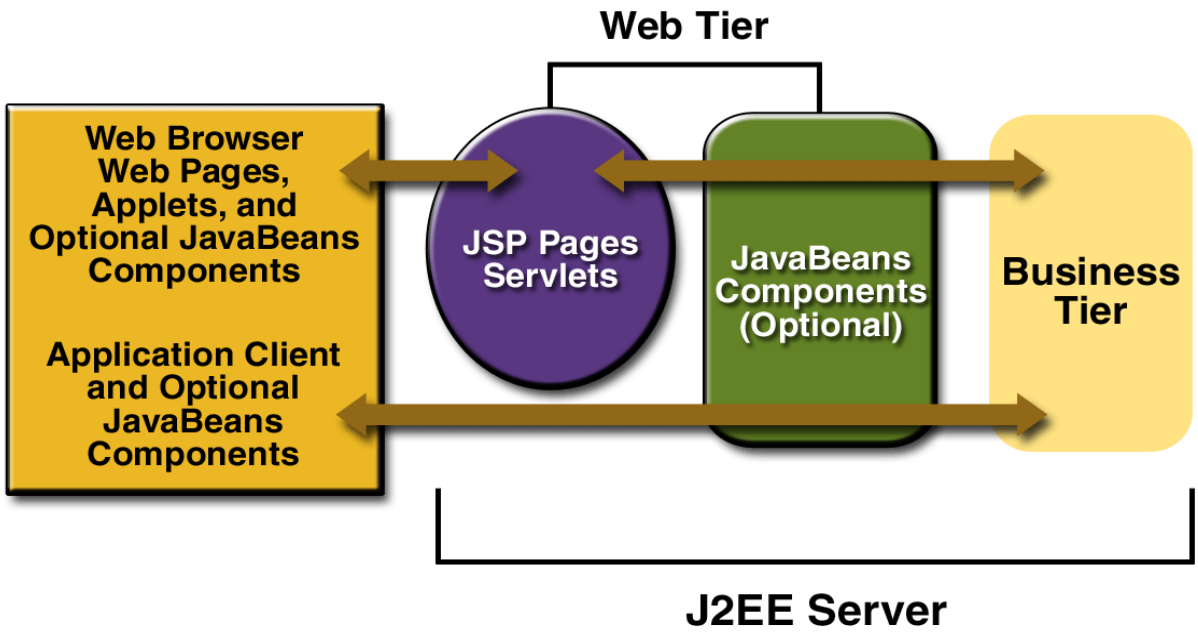
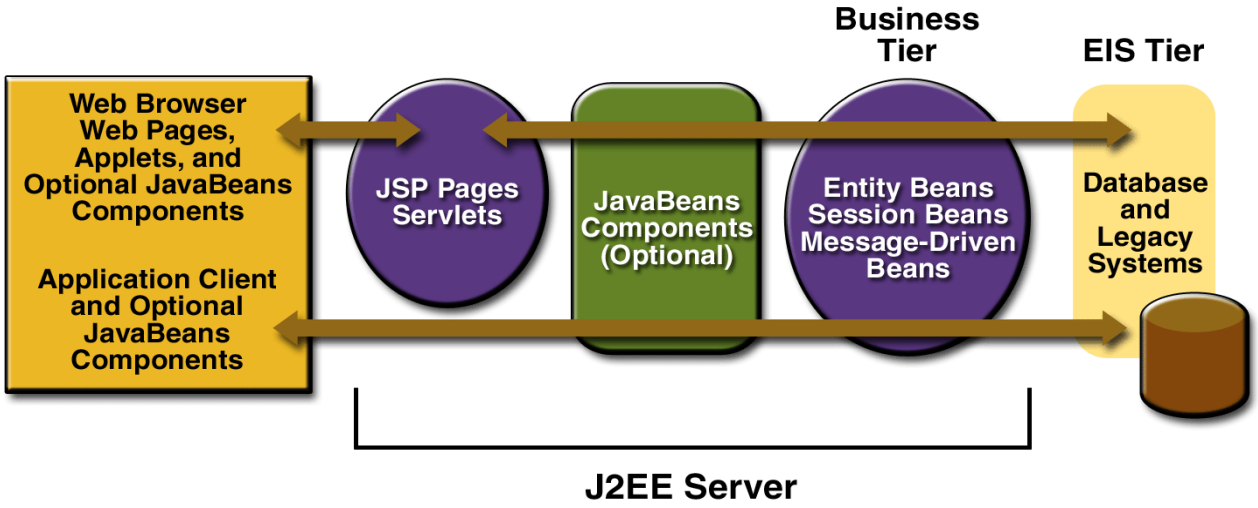


Figure 1-3 Web Tier and J2EE Application

BUSINESS COMPONENT

Business code, which is logic that solves or meets the needs of a particular business domain such as banking, retail, or finance, is handled by enterprise beans running in the business tier. An enterprise bean also retrieves data from storage, processes it (if necessary), and sends it back to the client program.



1. **Figure 1-4 Business and EIS Tiers**

There are three kinds of enterprise beans: session beans, entity beans, and message-driven beans. A *session bean* represents a transient conversation with a client. When the

client finishes executing, the session bean and its data are gone. In contrast, an *entity bean* represents persistent data stored in one row of a database table. If the client terminates or if the server shuts down, the underlying services ensure that the entity bean data is saved.

A *message-driven bean* combines features of a session bean and a Java Message Service (JMS) message listener, allowing a business component to receive JMS messages asynchronously. This tutorial describes entity beans and session beans.

ENTERPRISE INFORMATION SYSTEM TIER

The enterprise information system tier handles enterprise information system software and includes enterprise infrastructure systems such as enterprise resource planning (ERP), mainframe transaction processing, database systems, and other legacy information systems. J2EE application components might need access to enterprise information systems for database connectivity

JAVA SERVER PAGE (JSP)

Java Server Pages technology is the Java platform technology for building applications containing dynamic Web content such as HTML, DHTML and XML. The Java Server Pages technology enables the authoring of Web pages that create dynamic content easily but with maximum power and flexibility.

The Java Server Pages technology offers a number of advantages:

- *Write Once, Run Anywhere properties:*

The Java Server Pages technology is platform independent, both in its dynamic Web pages, its Web servers, and its underlying server components. You can author JSP pages on any platform,

run them on any Web server or Web enabled application server, and access them from any Web browser. You can also build the server components on any platform and run them on any server.

- *High quality tool support*

The Write Once, Run Anywhere properties of JSP allows the user to choose *best-of-breed* tools. Additionally, an explicit goal of the Java Server Pages design is to enable the creation of high quality portable tools.

- *Reuse of components and tag libraries*

The Java Server Pages technology emphasizes the use of reusable components such as: JavaBeans components, Enterprise JavaBeans components and tag libraries. These components can be used in interactive tools for component development and page composition. This saves considerable development time while giving the cross-platform power and flexibility of the Java programming language and other scripting languages.

- *Separation of dynamic and static content*

The Java Server Pages technology enables the separation of static content from dynamic content that is inserted into the static template. This greatly simplifies the creation of content. This separation is supported by beans specifically designed for the interaction with server-side objects.

- *Support for scripting and actions*

The Java Server Pages technology supports scripting elements as well as actions.

Actions permit the *encapsulation* of useful functionality in a convenient form that

can also be manipulated by tools; scripts provide a mechanism to *glue together* this

functionality in a per-page manner.

JSP ARCHITECTURE

JSPs are built on top of SUN's servlet technology. JSPs are essential an HTML page with special JSP tags embedded. These JSP tags can contain Java code. The JSP file extension is .jsp rather than .htm or .html. The JSP engine parses the .jsp and creates a Java servlet source file. It then compiles the source file into a class file; this is done the first time and this why the JSP is probably slower the first time it is accessed. Any time after this the special compiled servlet is executed and is therefore returns faster.

JAVA SCRIPT

JavaScript is a programming language that allows scripting of events, objects, and actions to create Internet applications. A website development environment that will allow the

creation of Interactive Web Pages. The coding techniques capable of accepting a client's requests and processing these requests.

The web site development environment should also provide the facility for 'validating' user input. With JavaScript, forms are a consideration in nearly every page you design. Capturing user requests is traditionally done via a 'form'. So the web site needs to have facilities to create forms. Text fields and textareas can dynamically change in response to user responses.

TOMCAT 5.0

Tomcat is a servlet container and Java Server Pages implementation it may be used stand alone ,or in conjunction with several popular web servers .

- Apache version 1.3 or later
- MS Internet Information Server ,version 4.0 or later
- MS personel web server, version 4.0 or later
- NetScape enterprise server , version 3.0 or later

Tomcat is a security update release.This release closes a whole that potentially allowed access to resourse protected by a <security constraint > in web.xml.

Installing and Running Tomcat 5.0

Tomcat requires a Java Runtime Environment (JRE).Conformant to JRE 1.1 or later including any Java2 platform system.If one wishes to develop applications you will need a java compiler , such as the one included in a java development kit 1.1 or later environment including JDKs conformant with Java2.

TESTING

TEST PLAN

A test plan implies a series of desired course of action to be followed in accomplishing various testing methods. The Test Plan acts as a blue print for the action that is to be followed. The software engineers create a computer program, its documentation and related data structures. The software developers is always responsible for testing the individual units of the programs, ensuring that each performs the function for which it was designed. There is an independent test group (ITG) which is to remove the inherent problems associated with letting the builder to test the thing that has been built. The specific objectives of testing should be stated in measurable terms. So that the mean time to failure, the cost to find and fix the defects, remaining defect density or frequency of occurrence and test work-hours per regression test all should be stated within the test plan.

The levels of testing include:

- Unit testing
- Integration Testing
- Data validation Testing
- Output Testing

6.1.1 UNIT TESTING

Unit testing focuses verification effort on the smallest unit of software design – the software component or module. Using the component level design description as a guide,

important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and uncovered scope established for unit testing. The unit testing is white-box oriented, and step can be conducted in parallel for multiple components. The modular interface is tested to ensure that information properly flows into and out of the program unit under test. The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm's execution. Boundary conditions are tested to ensure that all statements in a module have been executed at least once. Finally, all error handling paths are tested.

Tests of data flow across a module interface are required before any other test is initiated. If data do not enter and exit properly, all other tests are moot. Selective testing of execution paths is an essential task during the unit test. Good design dictates that error conditions be anticipated and error handling paths set up to reroute or cleanly terminate processing when an error does occur. Boundary testing is the last task of unit testing step. Software often fails at its boundaries.

Unit testing was done in Sell-Soft System by treating each module as separate entity and testing each one of them with a wide spectrum of test inputs. Some flaws in the internal logic of the modules were found and were rectified.

6.1.2 INTEGRATION TESTING

Integration testing is systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design. The entire program is tested as whole. Correction is difficult because isolation of causes is complicated by vast expanse of entire program. Once these errors are corrected, new ones appear and the process continues in a seemingly endless loop.

After unit testing in Sell-Soft System all the modules were integrated to test for any inconsistencies in the interfaces. Moreover differences in program structures were removed and a unique program structure was evolved.

6.1.3 VALIDATION TESTING OR SYSTEM TESTING

This is the final step in testing. In this the entire system was tested as a whole with all forms, code, modules and class modules. This form of testing is popularly known as Black Box testing or System tests.

Black Box testing method focuses on the functional requirements of the software. That is, Black Box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program.

Black Box testing attempts to find errors in the following categories; incorrect or missing functions, interface errors, errors in data structures or external data access, performance errors and initialization errors and termination errors.

6.1.4 OUTPUT TESTING OR USER ACCEPTANCE TESTING

The system considered is tested for user acceptance; here it should satisfy the firm's need. The software should keep in touch with perspective system; user at the time of developing and making changes whenever required. This done with respect to the following points

- Input Screen Designs,
- Output Screen Designs,

- Online message to guide the user and the like.

The above testing is done taking various kinds of test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested using that test data. While testing the system by which test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

Validation Checking:

At the culmination of integration testing, software is completely assembled as a package; interfacing errors have been uncovered and corrected, and a final series of software test-validation checks may begin. Validation can be defined in many ways, but a simple definition (Albeit Harsh) is that validation succeeds when software functions in a manner that can be reasonably expected by a customer. Software validation is achieved through a series of black-box tests to be conducted and a test procedure defines specific test cases that will be used in attempt to uncover errors in conformity with requirements. Both the plan and procedure are designed to ensure that all functional requirements are satisfied; all performance requirements are achieved; documentation is correct and human –Engineered and other requirements are met. Once the application was made free of all logical and interface errors , inputting dummy data to ensure that the software developed satisfied all the requirements of the user did validation checks .However , the data are created with the intent of determining whether the system will process them correctly .

In the proposed system, if the clients click the send button after selecting a file from his file list, then the system will show the confirmation message for sending files. Similarly if a client makes an attempt to download a file from the server file list, then also the system will show the confirmation message for downloading. This is how the data validations were made in the proposed system.

SCREENSHOT


PM Narendra Modi's Re
localhost:8096/CarPoolin
localhost:8096/CarPooling/
CARPOOL
carpooling
CLICK. RIDE. SAVE.
CARPOOL
Home
About-us
Contact-us
User Id
Password
Login
Register
Forgot UserId/Password
Copyright Reserved@LNC Infotech Pvt.Ltd
2:59 AM
11/29/2016


PM Narendra Modi's Rep
Welcome
localhost:8096/CarPooling/updateuser.jsp
carpooling
CLICK. RIDE. SAVE.
Welcome,shahzad
logout
Back to Main Menu
Update Details
Name
shahzad
Desired Password
123456
Email Id
shahzad.in.org@gmail.com
Mobile No
9654307241
Age
30
Gender
male
update
Copyright Reserved@LNC Infotech Pvt.Ltd
3:01 AM
11/29/2016


PM Narendra Modi's Rep
Welcome
localhost:8096/CarPooling/welcome_user.jsp
CARPOOL
carpooling
CLICK. RIDE. SAVE.
CARPOOL
Welcome,shahzad
logout
Get Your Car Registered!
Update Your Details
Update Your Car Details
Delete Your Car Details
Delete Your Account
Search For A Pool?
Copyright Reserved@LNC Infotech Pvt.Ltd
3:00 AM
11/29/2016


PM Narendra Modi's Re
Welcome
shahzad
localhost:8096/CarPooling/updateuser.jsp
CARPOOL
carpooling
CLICK. RIDE. SAVE.
CARPOOL
Welcome,shahzad
logout
Back to Main Menu
Update Details
Name
shahzad
Desired Password
123456
Email Id
shahzad.in.org@gmail.com
Mobile No
9654307241
Age
30
Gender
male
update
Copyright Reserved@LNC Infotech Pvt.Ltd
3:01 AM
11/29/2016


PM Narendra Modi's Rep
Welcome
shahzad
localhost:8096/CarPooling/updateuser_car.jsp
CARPOOL
carpooling
CLICK. RIDE. SAVE.
CARPOOL
Welcome,shahzad
logout
Back to Main Menu
Update Car Details
Car No
up11
Car Model
2009
Seats
4
Source
kat
Destination
sre
Departure Time
10
Return time
12
update
Copyright Reserved@LNC Infotech Pvt.Ltd
3:02 AM
11/29/2016


PM Narendra Modi's Rep
Welcome
shahzad
localhost:8096/CarPooling/search_pool.jsp
CARPOOL
carpooling
CLICK. RIDE. SAVE.
CARPOOL
Welcome,shahzad
logout
Back to Main Menu
Search for a Pool
Source
Destination
search
Name
Mobile No
Age
Gender
Source
Destination
Departure time
Return Time
Car Model
shahzad
9654307241
30
male
kat
sre
10
12
2009
Copyright Reserved@LNC Infotech Pvt.Ltd
3:03 AM
11/29/2016


PM Narendra Modi's Re
localhost:8096/CarPoolin
shahzad
localhost:8096/CarPooling/about-us.jsp
CARPOOL
carpooling
CLICK. RIDE. SAVE.
CARPOOL
Home
About-us
Contact-us
About Us
CARPOOLING
For Major Impact.
• Save Thousands of $$$
- Fuel Expense
- Vehicle Wear
- Parking and Tolls
• Reduce Pollution
• Reduce Greenhouse Gases
• Reduce Dependency on Foreign Oil
• Meet New People
• Reduce Stress From Driving
• Reduce Traffic
Carpooling is known as car-sharing, ride-sharing, lift-sharing and covoiturage when two or more people make the same commute ride in the same car or company vehicle to cut back on costs, traffic and air pollution, it is known as carpooling. Traffic is a major hindrance to commuters in large or poorly planned cities. Each individual may own their own car and so largely populated areas frequently suffer from air pollution or smog. Carpooling is an easy, inexpensive way for daily commuters to fight air pollution and also it reduces the number of vehicles on the road and the consumption of gasoline. Carpooling is common means of transport for many passengers, especially those who work in the same office or nearby offices. It is a good idea to think if you do not have your own vehicle and also if you do not want to experience the difficulty associated to traveling in public transport. A varying degree of formality and regularity is involved in carpooling arrangements and schemes. A variety of sources are responsible for motivation of carpooling. An individual who is environmentally conscious or others who want to enjoy the social or financial benefits may prefer carpooling. Additionally.
Copyright Reserved@LNC Infotech Pvt.Ltd
3:04 AM
11/29/2016

CODING

Car_registersodb file:-

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" import="p.*,java.sql.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String carno=request.getParameter("CarNumber");
String model=request.getParameter("Model");
String source=request.getParameter("Source");
String seats=request.getParameter("Seats");
String destination=request.getParameter("Destination");
String dptime=request.getParameter("DepartureTime");
String rttime=request.getParameter("ReturnTime");
String userid=(String)session.getAttribute("userid");
try
{
        Connection con=ConnectionProvider.getCon();
        PreparedStatement pst=con.prepareStatement("insert into carpool_usercardetails
values(?,?,?,?,?,?,?,?)");
        pst.setInt(1,Integer.parseInt(userid));
        pst.setString(2,carno);
```

```
        pst.setString(3,model);
        pst.setString(4,seats);
        pst.setString(5,source);
        pst.setString(6,destination);
        pst.setString(7,dptime);
        pst.setString(8,rttime);
        pst.executeUpdate();
        con.close();
        response.sendRedirect("welcome_user.jsp?msg=Your Car is Registered");
}
catch(Exception e)
{
System.out.println(e);
```

Car_registration file:-

```
<%@ include file="header1.html" %>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1" import="java.sql.*,p.*"%>

   <div style=" width:100px;font-size:15px;text-align:center;line-height:20px;color: blue;">
        Welcome,<%=session.getAttribute("username")%>
   </div>

   <div  style=" width:100px;font-size:15px;text-align:center;line-height:20px;color:
blue;margin-left:90%;">
        <a href="finallogout.jsp">logout</a>
   </div>
<div  style="width:150px;font-size:15px;text-align:center;line-height:20px;color: blue;">
        <a href="welcome_user.jsp">Back to Main Menu</a>
   </div>


        <center>
        <%
                String msg=request.getParameter("msg");
                if(msg!=null)
                out.println(msg);
        %>

        <%
```

```
try
{
String userid=(String)session.getAttribute("userid");
Connection con=ConnectionProvider.getCon();
PreparedStatement pst=con.prepareStatement("select * from carpool_usercardetails where user_id=?");
pst.setString(1,userid);

ResultSet rs=pst.executeQuery();
if(rs.next())
{
out.println("<h3>You Have Already Registered Your Car!</h3><br><br>");
out.println("<a href=updateuser_car.jsp>Update Car Details</a>");
}
else
{

%>
</center>
<h1 style="text-align:center;font-size:30px;color:brown">Car Registration Form</h1>
<form method="post" action="car_registertodb.jsp">
<table bgcolor="cyan" align="center" cellpadding="0px" cellspacing="10px">

<tr>
<td>Car No.</td><td><input type="text" name="CarNumber"/></td>

</tr>
<tr>
```

<td>Car Model</td><td><input type="text" name="Model"/></td>

</tr>

<tr>

<td>Seats In Car</td><td><input type="text" name="Seats"/></td>

</tr>

<tr>

<td>Source</td><td><input type="text" name="Source"/></td>

</tr>

<tr>

<td>Destination</td><td><input type="text" name="Destination"/></td>

</tr>

<tr>

<td>Departure Time</td><td><input type="text" name="DepartureTime"/></td>

</tr>

<tr>

<td>Return Time</td><td><input type="text" name="ReturnTime"/></td>

</tr>

<tr>

<td></td><td><input type="submit"/> <input type="reset" name="reset"/></td>

```
                                                    </tr>

            </table>

               </form>


            <%
            }
            con.close();

}
catch(Exception e)
{
System.out.println(e);
}
%>
                        <%@include file="footer.html"%>
```

Login.jsp file:-

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1" import="p.*,java.sql.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String userid=request.getParameter("userid");
String pass=request.getParameter("pass");
try
{
        Connection con=ConnectionProvider.getCon();
        PreparedStatement pst=con.prepareStatement("select * from carpool_userdetails
where user_id=? and user_password=?");
        pst.setString(1,userid);
        pst.setString(2,pass);

        ResultSet rs=pst.executeQuery();
        if(rs.next())
        {
                session.setAttribute("userid",userid);
```

```
                session.setAttribute("username",rs.getString("user_name"));
                response.sendRedirect("welcome_user.jsp");
        }
        else if("0".equals(userid) && "admin".equals(pass))
        {
                session.setAttribute("userid",userid);
                session.setAttribute("username","Admin");
                response.sendRedirect("welcome_admin.jsp");
        }
        else
        {
                response.sendRedirect("index.jsp?msg=Invalid User Id or Password");
        }
        con.close();

}
catch(Exception e)
{
System.out.println(e);
}
%>
</body>
</html>
```

Update_car.jsp:-

```
<%@ include file="header1.html" %>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1" import="java.sql.*,p.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Welcome</title>
</head>
<body>

   <div style=" width:100px;font-size:15px;text-align:center;line-height:20px;color: blue;">
         Welcome,<%=session.getAttribute("username")%>
   </div>

   <div  style=" width:100px;font-size:15px;text-align:center;line-height:20px;color:
blue;margin-left:90%;">
         <a href="finallogout.jsp">logout</a>
   </div>

   <div  style="width:150px;font-size:15px;text-align:center;line-height:20px;color: blue;">
         <a href="welcome_user.jsp">Back to Main Menu</a>
   </div>
```

```
<div  style=" width:200px;font-size:20px;text-align:center;line-
height:20px;color:green;margin-left:40%;">
        <%
          String msg=request.getParameter("msg");
          if(msg!=null)
          out.println(msg);
        %>
</div>


                              <%
try
{
          String userid=(String)session.getAttribute("userid");


          Connection con=ConnectionProvider.getCon();
          PreparedStatement pst=con.prepareStatement("select * from
carpool_usercardetails where user_id=?");
          pst.setString(1,userid);
          ResultSet rs=pst.executeQuery();
          if(rs.next())
          {
 %><h1 style="text-align:center;font-size:20px;color:brown">Update Car Details</h1>
                    <form action="updateusercartodb.jsp" method="post">
                  <table bgcolor="cyan" align="center" cellpadding="0px" cellspacing="10px">


                                                                  <tr>
                                                                        <td>Car
No</td><td><input type="text" name="carno" value="<%=rs.getString(2)%>"/></td>
```

```
                                                    </tr>
                                                    <tr>
                                                        <td>Car
Model</td><td><input type="text" name="carmodel" value="<%=rs.getString(3)%>"/></td>

                                                    </tr>
                                                    <tr>

        <td>Seats</td><td><input type="text" name="seats"
value="<%=rs.getString(4)%>"/></td>

                                                    </tr>
                                                <tr>

        <td>Source</td><td><input type="text" name="source"
value="<%=rs.getString(5)%>"/></td>

                                                    </tr>
                                                    <tr>

        <td>Destination</td><td><input type="text" name="destination"
value="<%=rs.getString(6)%>"/></td>

                                                    </tr>

                                                    <tr>
                                                        <td>Departure
Time</td><td><input type="text" name="outtime" value="<%=rs.getString(7)%>"/></td>

                                                    </tr>
                                                    <tr>
```

```
                                                    <td>Return
time</td><td><input type="text" name="intime" value="<%=rs.getString(8)%>"/></td>

                                                    </tr>

                                                    <tr>
                                                    <td></td><td><input
type="submit" value="update"/></td>

                                                    </tr>

                    </table>

                      </form>

  <%

          con.close();
          }
          else
                    {
                    out.println("<h3><center>You Have Not Registered Your Car
Yet!</center></h3><br><br>");
                    }
    }
    catch(Exception e)
    {
```

```
System.out.println(e);
}

%>

        </body>
</html>
<%@ include file="footer.html" %>
```

BIBILOGRAPHY

BOOKS:

- Charles Hampfed (2000) 'Instant Java Server Pages' University of Toronto
- Herbert Schildt (2000) 'Java Complete Reference' Tata McGraw Hill
- John Zukowski (2000) 'Mastering Java2' BPB Publications
- Jamie Jaworsky 'J2EE Bible' Techmedia
- Stefen Denninger 'Enterprise Java Beans-2.1' Author's Press
- Ian Somerville 'Software engineering'
- Rajeev mall 'Software engineering'
- Elmasri Navathe 'Fundamentals of database systems'

REFERENCE:

- www.theserverside.com
- www.java.sun.com