IMPLEMENTING THE FINANCIAL MANAGEMENT AND MARKETING SOFTWARE SYSTEM FOR FISHING INDUSTRY

BY

Mr.K.B.Neththikumara

(04/AS/109)

This thesis is submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Physical Sciences.

Department of Physical Science and Technology

Faculty of Applied Sciences

Sabaragamuwa University of Sri Lanka.

Belihuloya.

March 2009

DECLARATION

The content described in the thesis was practically implemented by me at the Asian Destination(pvt)Ltd and the Faculty of Applied Sciences under supervision of Mr.R.K.A.R. Kariapper and Mr. M.B.B.Kiridigoda and the report described on this thesis have not been submitted by any one for another degree.

Research Student

K.B.Neththikumara, ...Kelum......

Department of Physical Sciences, Signature

Faculty of Applied Sciences,

Sabaragamuwa University of Sri Lanka, ...07/05/2009.........

Belihuloya. Date

Certified by

Internal Supervisor

Mr.R.K.A.R. Kariapper, ...

Lecturer in Computer Science, Signature

Faculty of Applied Sciences,

Sabaragamuwa University of Sri Lanka, ...08/05/09....

Belihuloya. Date

External Supervisor

Mr. M.B.B.Kiridigoda , ...

Managing Director, Signature

Asian Destination(pvt)Ltd,

No.129/4/3, D.S.Senanayake Street, .07/05/2009

Kandy. Date

Dr. C.P.Udawatta,
Head,
Department of Physical Sciences,
Faculty of Applied sciences,
Sabaragamuwa University of Sri Lanka,
Belihuloya.

....
Signature

...2009.05.27...
Date

ACKNOWLEDGEMENT

I would like to thank to my internal supervisor Mr.R.K.A.R. Kariapper, for his continuous support in the Faculty of Applied Sciences, University of Sabaragamuwa Sri Lanka.

Programmer. Mr. M.B.B.Kiridigoda was always there to listen and to give advice. He is responsible for involving with me in implementing this Financial Management and Marketing software System for fishing industry. He taught how to ask questions and express my ideas. He showed me different ways to approach a research problem and the need to be persistent to accomplish any goal.

I express my sincere gratitude to all my team mates in the training and development team in Asian Destination(pvt)Ltd for providing me the opportunity to carry out my industrial training at Asian Destination(pvt)Ltd.

And I express my sincere gratitude to Prof.K.B.Pālipana, the Dean of faculty of Applied Sciences, University of Sabaragamuwa Sri Lanka and Dr.Udawatta, the Head of the Physical Department, faculty of Applied Sciences, University of Sabaragamuwa Sri Lanka.

My very special thank goes to Mr. T.N.Kurukulasooriya, who encourage me to take the very first step towards this project. He taught me Human-Computer Interaction techniques and how to do a usability study. His guidance showed me the right path for my goal. Thanks, Mr. Kurukulasooriya!

Finally I expressed my heartfelt gratitude towards the lectures for their cooperation throughout my study and my colleagues for their individual help and guidance at all time. Thanks for everyone who has helped me!

ABSTRACT

Archiving Solution for Fishing Industry

[1]K.B.Neththikumara, [2]R.K.A.Rifai Kariapper and [3] M.B.B.Kiridigoda

[1,2]Department of Physical Sciences, Faculty of Applied Sciences, Sabaragamuwa University

[3] Asian Destinations (Pvt)Ltd, No.129/4/3, D.S.Senanayake Street, Kandy

One of the key problems identified in using financial Software for fishing Industry. There are so many retail software using in fishing industry, but any type of retail software is not made for straight to the fishing Industry.

This is a financial system design for the Sri Lankan fishing industry. Targeted users are limited and selected. Primary target of this project is providing user friendly environment and make particular user to use this tool effectively. There will be various tools to involve and monitor all the trance actions regarding to the given privileges and to provide financial analyzing reports based on specific sections.

The software development core technologies are Java SE, MySQL, Enterprise Java Beans(EJB), Jasper reporting, Junit, UML. The requirements were based on the targeted user. Major requirements of this project were achieved by interviewing, the persons with knowledge of computer programs and knowledge to maintain fishing business cashbooks.

Then GUI was created according to the requirements and also consider about end users knowledge of understanding. Used netBeans 6.1 IDE to create GUI. After that separated database Schema was created by MySQL(database language based on Sql) databases. To verify this system, design class diagrams, sequence diagrams. Also used Jasper reporting and test through the Junit. Evaluative feedbacks were acquired in each meeting with external supervisor. Refinements and modifications were carried out after supervisor meeting to meet the client's requirements.

The components were tested individually and finally the integrated system was tested. At the end of the development process, the main objective of the project was achieved and the End User was satisfied with the functionalities, usability, security and reliability of the system.

Table of Contents

1.0 INTRODUCTION

Our core objective is to create and develop software to facilitate the requirements in Sri Lankan fishing industry cash book. In order to decide the most compatible programming language to create the software we have done a research among other programming languages. Which are mostly used in developing software?

The fishing industry in Sri Lanka includes any industry or activity concerned with taking, culturing, processing, preserving, storing, transporting, marketing or selling fish or fish products.

The fishing industry mainly ruled by three sectors

- The recreational sector: comprises enterprises and individuals associated for the purpose of recreation, sport or sustenance with fisheries resources from which products are derived that are not for sale.

- The traditional sector: comprises enterprises and individuals associated with fisheries resources from which aboriginal people derive products in accordance with their traditions.

- The commercial sector: comprises enterprises and individuals associated with wild-catch or aquaculture resources and the various transformations of those resources into products for sale. It is also referred to as the "seafood industry", although non-food items such as pearls are included among its products.

This software will be implemented as business software that helps business increase productivity or measure their productivity. The term covers a large variation of uses within the business environment, and can be categorized by using a small, medium and large matrix.

Targeted users are limited and selected. Primary target of this project is providing user friendly environment and make particular user to use this tool effectively. There will be various tools to involve and monitor all the trance actions regarding to the given privileges and to provide financial analyzing reports based on specific sections.

This project will be able to provide the environment due to the changes of the business arena and user requirements with plugging, Updates, Evolution copies, etc.

2.0 Project Plan

Software Development Life Cycle (SDLC)

1. Curtain Raiser

Like any other set of engineering products, software products are also oriented towards the customer. It is either market driven or it drives the market. Customer Satisfaction was the buzzword of the 80's. Customer Delight is today's buzzword and Customer Ecstasy is the buzzword of the new millennium. Products that are not customer or user friendly have no place in the market although they are engineered using the best technology. The interface of the product is as crucial as the internal technology of the product.

2. Market Research

A market study is made to identify a potential customer's need. This process is also known as market research. Here, the already existing need and the possible and potential needs that are available in a segment of the society are studied carefully. The market study is done based on a lot of assumptions. Assumptions are the crucial factors in the development or inception of a product's development. Unrealistic assumptions can cause a nosedive in the entire venture. Though assumptions are abstract, there should be a move to develop tangible assumptions to come up with a successful product.

3. Research and Development

Once the Market Research is carried out, the customer's need is given to the Research & Development division (R&D) to conceptualize a cost-effective system that could potentially solve the customer's needs in a manner that is better than the one adopted by the competitors at present. Once the conceptual system is developed and tested in a hypothetical environment, the development team takes control of it. The development team adopts one of the software development methodologies that is given below, develops the proposed system, and gives it to the customer.

The Sales & Marketing division starts selling the software to the available customers and simultaneously works to develop a niche segment that could potentially buy the software. In addition, the division also passes the feedback from the customers to the developers and the R&D division to make possible value additions to the product.

While developing software, the company outsourcers the non-core activities to other companies who specialize in those activities. This accelerates the software development process largely. Some companies work on tie-ups to bring out a highly matured product in a short period.

2.2 Popular Software Development Models

The following are some basic popular models that are adopted by many software development firms

System Development Life Cycle (SDLC) Model
Prototyping Model
Rapid Application Development Model
Component Assembly Model.. ect

Popular Software Development Models

Systems Development Life Cycle [Waterfall Model]

The waterfall model is a sequential software development model (a process for the creation of software) in which development is seen as flowing steadily downwards (like a waterfall) through the phases of

- requirements analysis,
- design,
- implementation,
- testing (validation),
- integration,
- Maintenance.

The origin of the term "waterfall" is often cited to be an article published in 1970 by Winston W. Royce (1929–1995), although Royce did not use the term "waterfall" in this article. Ironically, Royce was presenting this model as an example of a flawed, non-working model *(Royce 1970)*.

V Model

The V-model is a graphical representation of the systems development lifecycle. It summarizes the main steps to be taken in conjunction with the corresponding deliverables within computerized system validation framework.

The *VEE* is a process that represents the sequence of steps in a project life cycle development. It describes the activities and results that have to be produced during product. The left side of the *VEE* represents the decomposition of requirements, and creation of system specifications. The right side of the *V* represents integration of parts and their verification. V stands for "Verification and Validation"

2.2.3 Prototyping Model

This is a cyclic version of the linear model. In this model, once the requirement analysis is done and the design for a prototype is made, the development process gets started. Once the prototype is created, it is given to the customer for evaluation. The customer tests the package and gives his/her feed back to the developer who refines the product according to the customer's exact expectation.

2.2.4 Rapid Application Development (RAD) Model

The RAD model is a linear sequential software development process that emphasizes an extremely short development cycle. The RAD model is a "high speed" adaptation of the linear sequential model in which rapid development is achieved by using a component-based construction approach.

2.2.5 Component Assembly Model

Object technologies provide the technical framework for a component-based process model for software engineering. The object oriented paradigm emphasizes the creation of classes that encapsulate both data and the algorithm that are used to manipulate the data. If properly designed and implemented, object oriented classes are reusable across different applications and computer based system architectures. Component Assembly Model leads to software reusability. The integration/assembly of the already existing software components accelerates the development process. Nowadays many component libraries are available on the Internet. If the right components are chosen, the integration aspect is made much simpler

2.2.6 Spiral Model

The spiral model is a software development process combining elements of both design and prototyping-in-stages, in an effort to combine advantages of top-down and bottom-up concepts. Also known as the spiral lifecycle model, it is a systems development method (SDM) used in information technology (IT). This model of development combines the features of the prototyping model and the waterfall model. The spiral model is intended for large, expensive and complicated projects.

2.2.7 Chaos Model

The Chaos model is a structure of software development that extends the spiral model and waterfall model. The chaos model was defined by L.B.S. Raccoon.The chaos model notes that the phases of the life cycle apply to all levels of projects, from the whole project to individual lines of code.

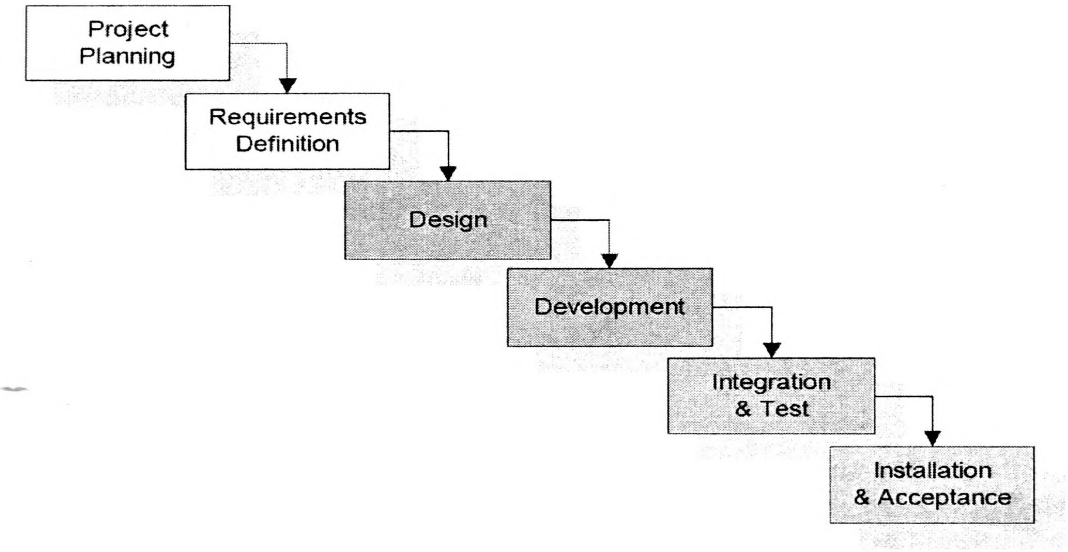
2.3 Popular Software Development Models in Details

2.3.1 Systems Development Life Cycle [Waterfall Model]

SDLC, the Software Development Life Cycle relates to models or methodologies that people use to develop systems, generally computer systems. Note: the acronym is sometimes thought of to represent Software Development Life Cycle and sometimes the process/model is simply referred to as the SLC. Computer systems have become more complex and usually (especially with the advent of SOA) link multiple traditional systems often supplied by different software vendors.

To manage this, a number of system development life cycle (SDLC) models have been created: waterfall, fountain and spiral build and fix, rapid prototyping, incremental, and synchronize and stabilize.

Small to medium database software projects are generally broken down into six
Stages:



The relationship of each stage to the others can be roughly described as a waterfall, where the outputs from a specific stage serve as the initial inputs for the following stage. During each stage,

Additional information is gathered or developed, combined with the inputs, and used to produce the stage deliverables. It is important to note that the additional information is restricted in scope;

"New ideas" that would take the project in directions not anticipated by the initial set of high-level requirements are not incorporated into the project. Rather, ideas for new capabilities or features that are out-of-scope are preserved for later consideration.

1. System Information Engineering and Modeling

As software is always of a large system (or business), work begins by establishing the requirements for all system elements and then allocating some subset of these requirements to software. This system view is essential when the software must interface with other elements such as hardware, people and other resources. System is the basic and very critical requirement for the existence of software in any entity. So if the system is not in place, the system should be engineered and put in place. In some cases, to extract the maximum output, the system should be re-engineered and spruced up. Once the ideal system is engineered or tuned, the development team studies the software requirement for the system.

2. Software Requirement Analysis

This process is also known as feasibility study. In this phase, the development team visits the customer and studies their system. They investigate the need for possible software automation in the given system. By the end of the feasibility study, the team furnishes a document that holds the different specific recommendations for the candidate system. It also includes the personnel assignments, costs, project schedule, target dates etc.... The requirement gathering process is intensified and focused specially on software. To understand the nature of the program(s) to be built, the system engineer or "Analyst" must understand the information domain for the software, as well as required function, behavior, performance and interfacing. The essential purpose of this phase is to find the need and to define the problem that needs to be solved.

3. System Analysis and Design

In this phase, the software development process, the software's overall structure and its nuances are defined. In terms of the client/server technology, the number of tiers needed for the package architecture, the database design, the data structure design etc... are all defined in this phase. A software development model is thus created. Analysis and Design are very crucial in the whole development cycle. Any glitch in the design phase could be very expensive to solve in the later stage of the software development. Much care is taken during this phase. The logical system of the product is developed in this phase.

4. Code Generation

the design must be translated into a machine-readable form. The code generation step performs this task. If the design is performed in a detailed manner, code generation can be accomplished without much complication. Programming tools like compilers, interpreters, debuggers etc... are used to generate the code. Different high level programming languages like C, C++, Pascal, Java are used for coding. With respect to the type of application, the right programming language is chosen.

5. Testing

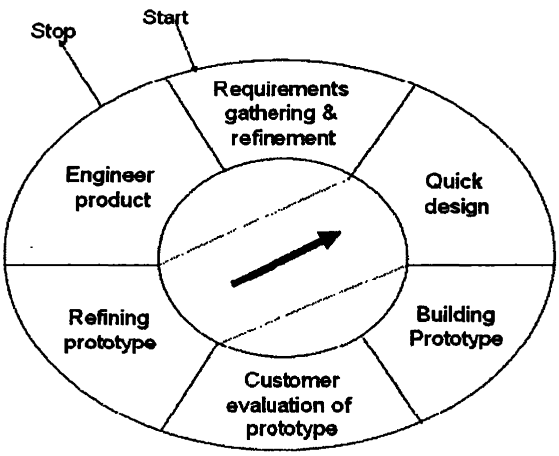
Once the code is generated, the software program testing begins. Different testing methodologies are available to unravel the bugs that were committed during the previous phases. Different testing tools and methodologies are already available. Some companies build their own testing tools that are tailor made for their own development operations.

6. Maintenance

The software will definitely undergo change once it is delivered to the customer. There can be many reasons for this change to occur. Change could happen because of some unexpected input values into the system. In addition, the changes in the system could directly affect the software operations. The software should be developed to accommodate changes that could happen during the post implementation period.

2.3.2. Prototyping Model

This is a cyclic version of the linear model. In this model, once the requirement analysis is done and the design for a prototype is made, the development process gets started. Once the prototype is created, it is given to the customer for evaluation. The customer tests the package and gives his/her feed back to the developer who refines the product according to the customer's exact expectation. After a finite number of iterations, the final software package is given to the customer. In this methodology, the software is evolved as a result of periodic shuttling of information between the customer and developer. This is the most popular development model in the contemporary IT industry. Most of the successful software products have been developed using this model - as it is very difficult (even for a whiz kid!) to comprehend all the requirements of a customer in one shot. There are many variations of this model skewed with respect to the project management styles of the companies. New versions of a software product evolve as a result of prototyping.



PROTOTYPE MODEL

Prototype Model Advantages

Creating software using the prototype model also has its benefits. One of the key advantages a prototype modeled software has is the time frame of development. Instead of concentrating on documentation, more effort is placed in creating the actual software. This way, the actual software could be released in advance. The work on prototype models could also be spread to others since there are practically no stages of work in this model. Everyone has to work on the same thing and at the same time, reducing man hours in creating a software. The work will even be faster and efficient if developers will collaborate more regarding the status of a specific function and develop the necessary adjustments in time for the integration.

Another advantage of having a prototype modeled software is that the software is created using lots of user feedbacks. In every prototype created, users could give their honest opinion about the software. If something is unfavorable, it can be changed. Slowly the program is created with the customer in mind.

o Prototype Model Disadvantages

Implementing the prototype model for creating software has disadvantages. Since its being built out of concept, most of the models presented in the early stage are not complete. Usually they lack flaws that developers still need to work on them again and again. Since the prototype changes from time to time, it's a nightmare to create a document for this software. There are many things that are removed, changed and added in a single update of the prototype and documenting each of them has been proven difficult.

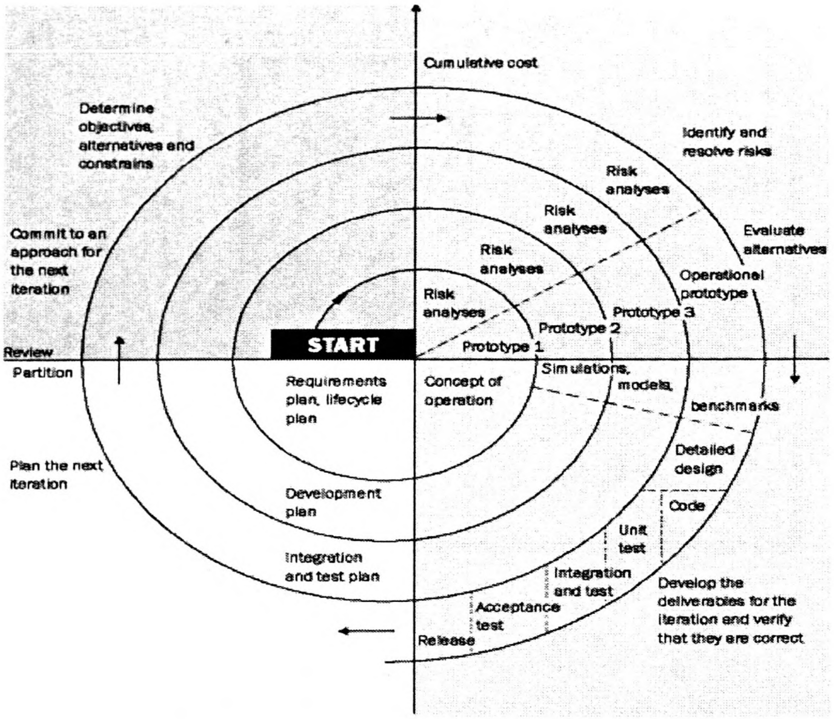
There is also a great temptation for most developers to create a prototype and stick to it even though it has flaws. Since prototypes are not yet complete software programs, there is always a possibility of a designer flaw. When flawed software is implemented, it could mean losses of important resources.

Lastly, integration could be very difficult for a prototype model. This often happens when other programs are already stable. The prototype software is released and integrated to the company's suite of software. But if there's something wrong the prototype, changes are required not only with the software. It's also possible that the stable software should be changed in order for them to be integrated properly.

2.3.3. Spiral Model

The steps in the spiral model can be generalized as follows:

1. The new system requirements are defined in as much detail as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
2. A preliminary design is created for the new system.
3. A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
4. A second prototype is evolved by a fourfold procedure:
    1. evaluating the first prototype in terms of its strengths, weaknesses, and risks;
    2. defining the requirements of the second prototype;
    3. planning and designing the second prototype;
    4. Constructing and testing the second prototype.
5. At the customer's option, the entire project can be aborted if the risk is deemed too great. Risk factors might involve development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer's judgment, result in a less-than-satisfactory final product.
6. The existing prototype is evaluated in the same manner as was the previous prototype, and, if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
7. The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
8. The final system is constructed, based on the refined prototype.
9. The final system is thoroughly evaluated and tested. Routine maintenance is carried out on a continuing basis to prevent large-scale failures and to minimize downtime.



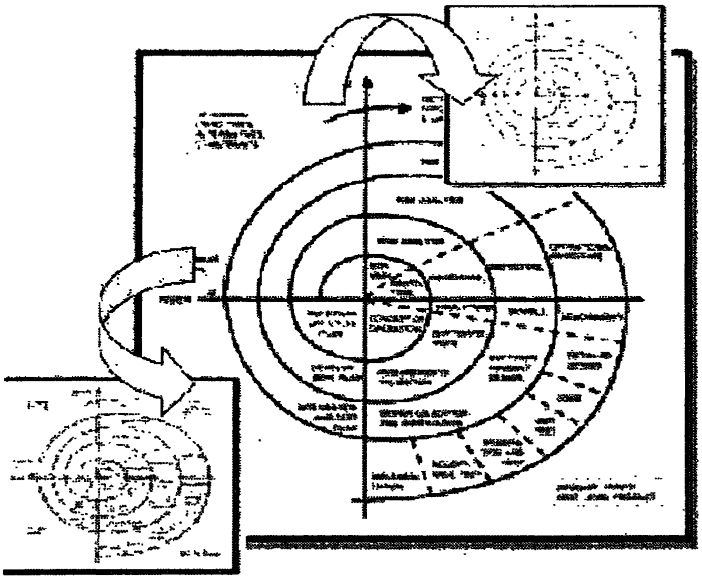
Cycle Requirements

- If alternatives or uncertainties are found they must be resolved.
- Risk-driven "subsetting" allows a mixture of other software process models, as necessary, until a high-risk situation is resolved.
  - Specification-oriented (Transform or Stage wise)
  - Prototype-oriented (Waterfall)
  - Automatic-transformation oriented (Transform)
  - Simulation-oriented (Evolutionary)

- Each cycle is completed by a review by the people concerned with the project.
- Plans for the next cycle should be introduced.
- With each succeeding level in the spiral the level of detail increases.

Overlapping Spirals

- Necessary for alternatives and parallel components.
- Stop everything until the break-away spiral is complete???
- Problems?



Advantages
□It balances resource expenditure.
□Doesn't involve separate approaches for software development and software maintenance.
□Provides a viable framework for integrated hardware-software system development.

Disadvantages
□Spiral model not yet complete (in 1988).
□Matching to contract software
– Internal projects have more freedom.
– Contract software demands total control and a full picture of the deliverables in advance.
□Relying on risk-assessment expertise.
□Need for further elaboration of spiral model steps.
– Milestones and specifications.
– Guidelines and checklists.

3.0 Deliverables

FISHER 2009 will provide some tangible and intangible objects to our valuable clients to drive the software more smoothly and gently. With these the client is able to involve in to a full user friendly environment.

3.1. Tangible substances.

- This project documentation
- User guide
- Software CD

3.2. Intangible substances

- The software
- The database

4.0 Project Scope

The areas we decided to cover in our software are:

- To create efficient and accurate Accounting system.
- To create user friendly and best benefited system for users.
- To create best logging system.

5.0 Research

5.1 Business research

Background

Our main task was to create and develop software for the financial sector of Fishing Industry. Therefore we have done a business research with few businesses located in Trincomalee area. The main reason to do a business research is to analyze the problems where fishing industry are facing in financial sector of fishing business. Even though there were many systems relate to the financial sector of fishing business, many of them weren't assured 100%. While on the research below are the main requirements we found.

Findings

- Requirement of a system to find profit of end of the month.
- Special requirement of them is, to print day today account reports of business
- Project will be able to provide the environment due to the changes of the business arena and user requirements with plugging, Updates, Evolution copies, etc.

Finally gathered all information and analyzed what are the main requirements in financial sector.

According to the requirements which have been gathered we have given our maximum benefits to fulfill requirements as results the profit finding system could be presented with the FISHER2009.

5.2 Technical Research

Programming Languages

Predecessor(s)	Year	Name	Chief developer, Company
1980s			
C, SIMULA 67	1980	C with classes	Stroustrup
BASIC, Compiler Systems, Digital Research	1980-1981	CBASIC	Gordon Eubanks
Smalltalk, C	1982	Objective-C	Brad Cox
BASICA	1983	GW-BASIC	Microsoft
Green	1983	Ada	CII Honeywell Bull
C with Classes	1983	C++	Stroustrup
BASIC	1983	True BASIC	Kemeny, Kurtz at Dartmouth College
COBOL	1983?	ABAP	SAP
sh	1984?	Korn Shell (*ksh*)	David Korn
Forth, Lisp	1984	RPL	Hewlett-Packard
ML	1984	Standard ML	
dBase	1984	CLIPPER	Nantucket
LISP	1984	Common Lisp	Guy Steele and many others
	1984	Redcode	A.K. Dewdney and D.G. Jones
Pascal	1985	Object Pascal	Apple Computer
dBase	1985	PARADOX	Borland
InterPress	1985	PostScript	Warnock
BASIC	1985	QuickBASIC	Microsoft
BASIC	1986	GFA BASIC	Frank Ostrowski
	1986	Miranda	David Turner at University of Kent
	1986	LabVIEW	National Instruments
SIMULA 67	1986	Eiffel	Meyer
	1986	Informix-4GL	Informix
C	1986	PROMAL	
INFORM	1986	CorVision	Cortex
Smalltalk	1987	Self (concept)	Sun Microsystems Inc.

	1987	HyperTalk	Apple
Ada, C++, Lisp	2000	XL	Christophe de Dinechin
C, C++, Java, Delphi	2000	C#	Anders Hejlsberg at Microsoft(ECMA)
C, C++, Java, PHP, Python, Ruby, Scheme	2000	Ferite	Chris Ross
Java	2001	AspectJ	Xerox PARC
Self, NewtonScript	2002	Io	Steve Dekorte
C#, ML, MetaHaskell	2003	Nemerle	University of Wrocław
Joy, Forth, Lisp	2003	Factor	Slava Pestov
Smalltalk, Java, Haskell, Standard ML, OCaml	2003	Scala	Martin Odersky
BASIC	2004	FreeBASIC	Andre Victor
Mobile Development	2004	WinDev Mobile	PC Soft
*	2004	Subtext	Jonathan Edwards
Python, C#	2004	Boo	Rodrigo B. de Oliveira
Object Pascal, C#	2004	Oxygene (formerly Chrome)	RemObjects Software
Java	2004	Groovy	James Strachan
BASIC	2004	ThinBasic	Eros Olmi thinBasic community
Haskell	2006	Links	Phil Wadler, University of Edinburgh
*	2006	Kite	Mooneer Salem
C#, ksh, Perl, CL, DCL, SQL	2006	Windows PowerShell	Microsoft
APEX	2007	APEX	Salesforce.com
C#	2007	Vala	GNOME
C, R	2008	PCASTL	Philippe Choquette

5.2.1. Java

Java, the language, is a high-level object-oriented programming language, influenced in various ways by C, C++, and Smalltalk, with ideas borrowed from other languages as well (see O'Reilly's History of Programming Languages). Its syntax was designed to be familiar to those familiar with C-descended "curly brace" languages, but with arguably stronger OO principles than those found in C++, static typing of objects, and a fairly rigid system of exceptions that require every method in the call stack to either handle exceptions or declare their ability to throw them. Garbage collection is assumed, sparing the developer from having to free memory used by obsolete objects.

One of Java's more controversial aspects--widely accepted at the time of its release but increasingly criticized today--is its incomplete object-orientation. Specifically, Java primitives such as int, char, boolean, etc. are not objects, and require a completely different treatment from the developer: as int is not a class, you cannot subclass and declare new methods on it, cannot pass it to a method that expects a generic Object, and so on. The inclusion of primitives increases Java performance, but at the arguable expense of code clarity, as anyone who's had to work with the so-called "wrapper classes" (Integer, Character, and Boolean) will attest. Java 5.0 introduces an "autoboxing" scheme to eliminate many uses of the wrapper classes, but in some ways it obscures what is really going on.

Philosophically, Java is a "fail early" language. Because of its syntactic restrictions, many programming failures are simply not possible in Java. With no direct access to pointers, pointer-arithmetic errors are non-existent. Using an object as a different type than what it was originally declared to be requires an explicit cast, which gives the compiler an opportunity to reject illogical programming, like calling a String method on an Image.

Many Java enterprise frameworks require the use of configuration files or deployment descriptors, typically written in XML, to specify functionality: what class handles a certain HTTP request, the order of steps to execute in a rule engine, etc. In effect, they have to go beyond the language to implement their functionality. Critics point out that this has the perverse effect of not only escaping Java's compiler checks, but also that a developer can no longer determine how a program will operate just by looking at its source code. Java 5.0 adds annotations to the language, which allows the tagging of methods, fields, and classes with values that can then be inspected and operated on at runtime, usually through reflection. Many programmers like annotations because they simplify tasks that might otherwise be addressed by deployment descriptors or other means. But again, they can make it difficult to understand Java code, as the presence or absence of an annotation may affect how the code is executed, in ways that are in no way obvious from the annotation.

Despite these criticisms, Java is generally understood to be the most popular general-purpose computing language in use today. It is a widely used standard in enterprise programming, and in 2005, it replaced C++ as the language most used by projects on SourceForge. What it has going for it is immense: free tools (on multiple platforms: Linux, Windows, Solaris, and Mac can all compile and execute Java apps), a vast base of knowledge, and a large pool of readily available developers.

The Java language hits a specific point in the tradeoff between developer productivity and code performance: CPU cycles keep getting cheaper, developers largely don't, so it is perhaps inevitable to accept another layer of abstraction between the developer and the execution of CPU opcodes, if it allows the developer to create better software faster. In fact, critics of Java's productivity, such as Bruce Tate in Beyond Java, may simply be observing this trend continuing past Java to a new sweet spot that further trades performance for developer productivity.

The Java Platforms

Java is generally thought of in terms of three platforms: Standard Edition (SE), Enterprise Edition (EE), and Micro Edition (ME). Each describes the combination of a language version, a set of standard libraries, and a virtual machine (see below) to execute the code. EE is a superset of SE--any EE application can assume the existence of all of the SE libraries--and EE's use of the language is identical to SE's.

Because of the limitations of small devices like phones and set-top boxes, Java Micro Edition differs significantly from its siblings. It is not a subset of SE (as SE is of EE), as some of its libraries exist only in Micro Edition. Moreover, ME eliminates some language features, such as the float primitive and Float class, reflecting the computing limitations of the platforms it runs on. Requiring different tools than SE and EE, and with profound differences in devices that makes code portability far less realistic in the micro space, many Java developers see ME as utterly alien.

The Java Virtual Machine

At some point, Java source needs to become platform-native executable code. This typically requires a two-step process: the developer compiles his or her source into Java bytecode, and then a Java Virtual Machine (JVM) converts this into native code for the host platform. This latter step originally was performed by interpretation--taking each JVM instruction and converting it on the fly to one or more native instructions. Later, just-in-time (JIT) compilers converted all of a Java program from JVM bytecode to native code as the program started up. In the modern era, there are multiple approaches. Sun's HotSpot compiler starts by interpreting code and profiling it at runtime, compiling and optimizing those parts that are found to be most critical to the program's operation. The "mixed mode interpreter" of IBM's JVMs works much the same way. These approaches avoid the startup performance hit entailed by JITing the entire program, but means that performance arrives over time, as critical code sections are located and optimized. Long-running server processes are well-served by this approach, client applications less so.

As is the case with primitives, the two-step compile cycle of Java now starts to look like a premature optimization to some critics. If you're going to wait until runtime to compile from Java bytecode to native, they ask, why not save the developer a step by interpreting not Java bytecode, but Java source? As Tate notes in Beyond Java, "Java is not the simplest of languages. Nor is it friendly to very short iterations ... Other languages let you move from one change to the next without a cumbersome compile/deploy cycle."

The JVM Without Java

Indeed, one of Tate's key criteria in finding potential successors to Java's success is the idea that "the next commercially successful language should have a version that runs in the JVM. That would help a language overcome many obstacles, both political and technical." He points out that a VM approach gives you security ("if you can secure the virtual machine, it's much easier to secure the language"), portability, interoperability, and extensibility. With the JVM having effectively solved these problems, a new language wouldn't need its own VM if it can simply run in the JVM that is already on millions of computers.

In many ways, this is already happening. Writing interpreters for scripting languages in Java effectively brings these languages to the JVM, like Rhino for JavaScript, Jython for Python, or JRuby for Ruby.

But it's also possible to bypass the Java language altogether and go straight to the JVM level. There are already C-to-JVM bytecode compilers, such as the commercial Axiomatic Multi-Platform C, which provides a subset of ANSI C. Furthermore, the growth of Java bytecode manipulation with tools such as ASM and Apache BCEL allow Java applications to create executable classes at runtime. This is no longer Java, but effectively a form of assembly language programming for the JVM.

Perhaps appreciating the desire to run non-Java code on the JVM, a new JSR, "Supporting Dynamically Typed Languages on the JavaTM Platform" (JSR 292), has recently been introduced, specifying a new bytecode that would make the JVM better suited to running languages without static type information.

Java Without the JVM

You can also turn the tables the other way, and run Java without a JVM. After all, at some point, Java source becomes bytecode, which in turn becomes native code--and nobody said you can't do it all at once. The GNU Compiler for Java (GCJ) allows for a one-time, up-front compilation of Java source into an executable for a single platform. While incomplete--it does not support the Abstract Windowing Toolkit (AWT), thus making it unsuitable for AWT or Swing GUI programming--there's enough there to compile server-side and command-line applications.

This process has one obvious downside: cross-platform code becomes bound to a single platform in one step. Moreover, the static compilation is not an automatic trump of HotSpot's dynamic compilation--the author once worked on a project where the performance gain from GCJ was found to be less than five percent beyond the HotSpot version. Still, GCJ can solve important problems, like deploying a runnable Java application without having to worry whether a JVM is available or running a specific version.

Advantages of JAVA

JAVA offers a number of advantages to developers.

Java is simple: Java was designed to be easy to use and is therefore easy to write, compile, debug, and learn than other programming languages. The reason that why Java is much simpler than C++ is because Java uses automatic memory allocation and garbage collection where else C++ requires the programmer to allocate memory and to collect garbage.

Java is object-oriented: Java is object-oriented because programming in Java is centered on creating objects, manipulating objects, and making objects work together. This allows you to create modular programs and reusable code.

Java is platform-independent: One of the most significant advantages of Java is its ability to move easily from one computer system to another.

The ability to run the same program on many different systems is crucial to World Wide Web software, and Java succeeds at this by being platform-independent at both the source and binary levels.

Java is distributed: Distributed computing involves several computers on a network working together. Java is designed to make distributed computing easy with the networking capability that is inherently integrated into it.

Writing network programs in Java is like sending and receiving data to and from a file. For example, the diagram below shows three programs running on three different systems, communicating with each other to perform a joint task.

Java is interpreted: An interpreter is needed in order to run Java programs. The programs are compiled into Java Virtual Machine code called bytecode.

The bytecode is machine independent and is able to run on any machine that has a Java interpreter. With Java, the program need only be compiled once, and the bytecode generated by the Java compiler can run on any platform.

Java is secure: Java is one of the first programming languages to consider security as part of its design. The Java language, compiler, interpreter, and runtime environment were each developed with security in mind.

Java is robust: Robust means reliable and no programming language can really assure reliability. Java puts a lot of emphasis on early checking for possible errors, as Java compilers are able to detect many problems that would first show up during execution time in other languages.

Java is multithreaded: Multithreaded is the capability for a program to perform several tasks simultaneously within a program. In Java, multithreaded programming has been smoothly integrated into it, while in other languages, operating system-specific procedures have to be called in order to enable multithreading. Multithreading is a necessity in visual and network programming.

Disadvantages of JAVA

Performance: Java can be perceived as significantly slower and more memory-consuming than natively compiled languages such as C or C++.

Look and feel: The default look and feel of GUI applications written in Java using the Swing toolkit is very different from native applications. It is possible to specify a different look and feel through the pluggable look and feel system of Swing.

Single-paradigm language: Java is predominantly a single-paradigm language. However, with the addition of static imports in Java 5.0 the procedural paradigm is better accommodated than in earlier versions of Java.

FEATURES

Here we list the basic features that make Java a powerful and popular programming language:

- **Platform Independence**
  - The *Write-Once-Run-Anywhere* ideal has not been achieved (tuning for different platforms usually required), but closer than with other languages.
- **Object Oriented**
  - Object oriented throughout - no coding outside of class definitions, including main ().
  - An extensive class library available in the core language packages.

- **Compiler/Interpreter Combo**
  - Code is compiled to bytecodes that are interpreted by a Java virtual machines (JVM)
  - This provides portability to any machine for which a virtual machine has been written.
  - The two steps of compilation and interpretation allow for extensive code checking and improved security.

- **Robust**
  - Exception handling built-in, strong type checking (that is, all data must be declared an explicit type), local variables must be initialized.

- **Several dangerous features of C & C++ eliminated:**
  - No memory pointers
  - No preprocessor
  - Array index limit checking

- **Automatic Memory Management**
  - o Automatic garbage collection - memory management handled by JVM.

- **Security**
  - o No memory pointers
  - o A program runs inside the virtual machine sandbox.
  - o Array index limit checking
  - o Code pathologies reduced by

▪ *bytecode verifier* - checks classes after loading
▪ *class loader* - confines objects to unique namespaces. Prevents loading a hacked "java.lang.SecurityManager" class, for example.
▪ *security manager* - determines what resources a class can access such as reading and writing to the local disk.

- **Dynamic Binding**
  - o The linking of data and methods to where they are located, is done at run-time.
  - o New classes can be loaded while a program is running. Linking is done *on the fly*.
  - o Even if libraries are recompiled, there is no need to recompile code that uses classes in those libraries.

This differs from C++, which uses static binding. This can result in *fragile* classes for cases where linked code is changed and memory pointers then point to the wrong addresses.

- **Good Performance**
  - o Interpretation of bytecodes slowed performance in early versions, but advanced virtual machines with adaptive and just-in-time compilation and other techniques now typically provide performance up to 50% to 100% the speed of C++ programs.

- **Threading**
  - o Lightweight processes, called threads can easily be spun off to perform multiprocessing.
  - o Can take advantage of multiprocessors where available
  - o Great for multimedia displays.

- **Built-in Networking**
  - o Java was designed with networking in mind and comes with many classes to develop sophisticated Internet communications.

Why Software Developers Choose Java

Java has been tested, refined, extended, and proven by a dedicated community. And numbering more than 6.5 million developers, it's the largest and most active on the planet. With its versatility, efficiency, and portability, Java has become invaluable to developers by enabling them to:

- Write software on one platform and run it on virtually any other platform
- Create programs to run within a Web browser and Web services
- Develop server-side applications for online forums, stores, polls, HTML forms processing, and more
- Combine applications or services using the Java language to create highly customized applications or services
- Write powerful and efficient applications for mobile phones, remote processors, low-cost consumer products, and practically any other device with a digital heartbeat

Differences and why Java becomes “FISHER 2009” software main language

Java is easier because...

- Java checks for errors. For example, Java checks subscripts to make sure they are in the correct range.
- Java does things for the programmer. There are a huge number of things that Java has already written for programmer. For example, expandable arrays, many data structures, etc. In C it would take a very long time to write and debug these things by ourselves.
- Java doesn't have the most dangerous things. The things in C which cause the most program errors are pointers, pointer arithmetic, and memory management. Java has replaced these with much, much safer things: references, subscription, and garbage collection.

The reason that everyone is very enthusiastic about Java is because it is easier (faster, cheaper,) to produce good programs

Object Oriented programming

• Classes form the basis for Object Oriented Programming (OOP). Java classes are like C++ class, which are something like a C struts that includes both data fields and functions. Objects are created when memory is allocated for one of these class "struts". The basic idea is simple.

Java support for large programming project

For small programming projects it's sufficient to use functions, sometimes in separate source files. Large programming projects need more control over the structure and visibility of the program elements. Java provides this control in the form of *classes*, *packages*, and *interfaces*, along with a number of ways to control who can see what. In FISHER 2009, a lot of packages and interfaces have been used so it was more compatible to handle java than C.

Class Libraries

Sometimes someone says that Java is a small language, smaller than C++, and perhaps smaller than C. This may be true if you ignore one of the most important things: the class libraries. There are a huge number of *methods* (the special OOP word for function) in the thousands of classes that are grouped in *packages*. Packages and OOP are the reasons that it's possible to have such large libraries.

Java is harder because ...

- Java is more powerful and can do much more than C. For example, C doesn't have a graphical user interface (GUI), and C doesn't have any way to do object-oriented programming (OOP). It's possible to write in Java in a C style, avoiding the new powerful features of Java. But that is foolish.
- Java either checks for errors, or makes you check for errors. C lets you do many things that would cause errors (for example, convert strings to integers, or do I/O), but doesn't make you write code to handle the errors. Java makes you write try...catch statements around things that might cause problems.

Portability and types

- Because of the lack of complete type definitions, moving a C program from one machine to another can be a giant headache, or even impossible. For example, to preserve the range of an integer variable you might have to change shorts to ints, but then you also have to change the corresponding format specifiers, union declarations, bit field declarations, shift operators, etc. This is only one example of the tremendous number of portability problems in C.
- The Java types are well defined, there are none of these problems.

Third Major Trend: Visual Programming and Components

As noted previously, visual programming and the use of components is one of the pioneering strengths of Java and Visual Basic. OO methods make GUI and general program development using components even easier to do. But these techniques have been spawned by a two very real needs. N-tier distributed processing has brought two real benefits - isolated islands of information and inefficient use of data and program resources are being eliminated. With intranets and online processing, users can reach into every nook and cranny of an organization to get at the data and information they need to solve a problem. But the cost as noted is the much greater complexity of distributed programs - even ones that use web browser front ends. Hence the move to outsourcing and/or packaged programs.

However a second pressing need is to develop software so much faster. For over 30 years two of the major complaints against IT departments has been that systems cannot be developed fast enough (the 2-3 year backlog of IT projects) and then when delivered changes and updates to programs cannot be done on a timely basis. But with the huge repository of computer systems built up over the last 30 years and with the availability of complete backbone office systems from major ERP vendors, programming has been turned upside down. Now IT staffs are being told "No" about re-inventing the IT wheel - it takes too long and is fraught with high risk of complete failure (50% failure rate for large client/server projects according to the Standish Reports). Reuse and interfacing to new ERP backbones as well as legacy systems are the new marching orders for IT development teams at small as well as large businesses. In effect, all programming has become maintenance programming. Very few programs are built from scratch. Programs have to interface with legacy databases or web systems. Many have to link to or just become a customized user called procedure of some ERP or other packaged program. What better way to create these component systems? Bingo - visual programming and component development with Java and Visual Basic

The Essential Advantages of Java and Visual Basic...

Without a doubt one of the essential advantages of both Java and Visual Basic is that programmers can develop a wide range of programs so much faster than other language. So called 4GLs (fourth generation languages) like Focus, PowerBuilder and Uniface once held an advantage in ease and speed of development (but at the cost of poor runtime performance). But 3GL-3rd Generation Languages and especially Java and Visual Basic with their visual programming, components and use of clever wizards or third party design tools have matched if not surpassed 4GL for speed of development and have superior runtime performance. Relative to other 3GLs such as Ada, C/C++, COBOL, FORTRAN, Pascal and others; Java and Visual Basic have three compelling advantages:

-new, uniform and comprehensive APIs for every aspective of web, client/server, and n-tier programming;

-several modes of deployment including .EXEs, components, servlets and applets;
-superior visual programming development environs with many 3rd party suppliers of tools and components.

And the runtime performance of both Java and Visual Basic have steadily improved to the point that they can approach within 10-30% of the fastest programs developed in C/C++, Cobol, Fortran or Pascal. Finally, Java in particular but also Visual Basic can be used to deliver web programs.

On the Web, Java and Visual Basic can be used as servlets - web applications sitting on a server and sending down dynamically created HTML and scripts to your browser. Java with its great cross platform portability can also be used to develop applets that run directly on any PC or client browser. With new ADSL and cable modems running 20 times faster than current 56K modems, look for even more exciting (and secure) Java applets coming to your web browser. In sum, both Java and Visual Basic have come to the fore as programming languages because they were built to fit the mold for new system development - rapid development through visual IDEs and comprehensive SDK/APIs to meet all the latest programming requirements, especially the Web. Perhaps even more important are OO methods and componentized code which can be stand alone or linked to existing programs and data sources in a variety of ways - servlet, downstream application, application server or web applet. It is this versatility in deployment that makes both languages so attractive.

Over the next 4-5 years the proliferation in programming languages should subside. This situation has persisted so long as client/server and n-tier processing needs were not being well served. But the current balance of comprehensive APIs, rapid development for a variety of deployment modes and competitive runtime performance make Java and Visual Basic compelling choices for many business applications. So expect programming languages to gravitate towards the specialties that they serve well: Prolog and some LISP variants for AI-Artificial Intelligence, Fortran for intensive engineering/numeric analysis systems; C/C++ for commercial software development, Cobol for transaction processing and most notably Java and Visual Basic leading the way on the new web and component based systems that acts as links and linchpins among backoffice applications.

Stability and Reliability

Traditionally, programming languages have had very good records for stability and reliability for both the development systems and the program code they produce. However, in the rush to get newest versions and features out the door, both Java and Visual Basic let down developers. Part of this may be inevitable - rapid changes in web and programming standards mean that some code and APIs gets obsolete - as for example when many AWT classes were quickly superseded by the new event-listener classes and Swing GUI components and interfaces. In VB a similar pattern has seen two or three data access methods proliferate into over a dozen. Unfortunately, performance, reliability, and functionality trade-offs among the competing access methods make choosing an approach in VB, even with the new OLE-DB standard, a daunting task. But perhaps the worst problem is the lacunae of bugs in both languages.

To an extent bugs come with rapid change and huge APIs. But for the last 3 versions of VB, users have had to wait for two or even three service packs to be able to get reliable code for some new features either in the development system or in the language. See the editorial in April 1999 Visual Basic Programmer's Journal "Five Things You Hate about VB6" for the latest episode on VB bugs. Yes, most of the old code works. Yes, most of the bugs are clustered around new features. No, most users do not want to develop with what is in effect beta code and features. Want to kill a function or feature in a language - make it buggy so developers have to spend extra hours and days proving that the bug is in the language not their code.

Java is somewhat better. But in the rush to get various releases of JDK 1.x out the door serious bugs have crept into some of the new APIs. Also compatibility of the GUI routines on various OS platforms has lead to pungent humor - Java is the language you get to write once and debug everywhere. Now it is important to point out that these are not catastrophic bugs by any means. Ninety five percent or more of the functionality of both languages is rock solid. However when you are developing using some of the 5 percent of features that are a bug infested swamp (the lacunae),

In their defense, both Microsoft and Sun have rededicated themselves to producing more reliable code. Microsoft, for example, has made service packs easier to access and more quickly available. Sun has released later versions of the1.x JDK in smaller and more reliable chunks while delaying key features like some EJBs and Hotspot technology for many months. But there is a lot at stake. Other languages like C/C++ or ObjectPascal/Delphi or the new versions of Cobol and PowerBuilder have a better record for reliability. In effect, by delivering buggy code and/or development systems Java and Visual Basic may jeopardize their own critical advantage - speed of development.

c# is one of the most visible aspects of Microsoft's .NET initiative. Developers commonly see the world through the lens of their language, and so C# can be their main aperture into .NET. Misconceptions abound, however. In this article, I'd like to address the confusions I hear most often about this new tool.

Summarization

When we work through those details which have been included in above section it can be easy to understand that VB language family is being changed rapidly the key point is when this change is happening the core concepts , objectives & even the technical syntax have been changed therefore the one who work with VB languages must be changed their technical culture and the core objectives according to the trends but, when we consider about java it has very strong and stable culture and core concepts. So the programmers who involve or work with java no need to rapidly changed according to the trends, only need to vast their knowledge throughout the new implementations. Therefore fro creating FISHER2009 java has been used over VB languages.

5.2.2 SQL

SQL is a standard interactive and programming language for querying and modifying data and managing databases. Although SQL is both an ANSI and an ISO standard, many database products support SQL with proprietary extensions to the standard language. The core of SQL is formed by a command language that allows the retrieval, insertion, updating, and deletion of data, and performing management and administrative functions. SQL also includes a Call Level Interface (SQL/CLI) for accessing and managing data and databases remotely.

The first version of SQL was developed at IBM by Donald D. Chamberlin and Raymond F. Boyce in the early 1970s. This version, initially called SEQUEL, was designed to manipulate and retrieve data stored in IBM's original relational database product, System R. The SQL language was later formally standardized by the American National Standards Institute (ANSI) in 1986. Subsequent versions of the SQL standard have been released as International Organization for Standardization (ISO) standards.

Originally designed as a declarative query and data manipulation language, variations of SQL have been created by SQL database management system (DBMS) vendors that add procedural constructs, control-of-flow statements, user-defined data types, and various other language extensions. With the release of the SQL:1999 standard, many such extensions were formally adopted as part of the SQL language via the SQL Persistent Stored Modules (SQL/PSM) portion of the standard.

Common criticisms of SQL include a perceived lack of cross-platform portability between vendors, inappropriate handling of missing data (Null (SQL)), and unnecessarily complex and occasionally ambiguous language grammar and semantics.

History

During the 1970s, a group at IBM's San Jose research center developed the System R relational database management system, based on the model introduced by Edgar F. Codd in his influential paper, A Relational Model of Data for Large Shared Data Banks.Donald D. Chamberlin and Raymond F. Boyce of IBM subsequently created the Structured English Query Language (SEQUEL) to manipulate and manage data stored in System R. The acronym SEQUEL was later changed to SQL because "SEQUEL" was a trademark of the UK-based Hawker Siddeley aircraft company.

The first non-commercial non-SQL RDBMS, Ingres, was developed in 1974 at the U.C. Berkeley. Ingres implemented a query language known as QUEL, which was later supplanted in the marketplace by SQL.

In the late 1970s, Relational Software, Inc. (now Oracle Corporation) saw the potential of the concepts described by Codd, Chamberlin, and Boyce and developed their own SQL-based RDBMS with aspirations of selling it to the U.S. Navy, CIA, and other government agencies. In the summer of 1979, Relational Software, Inc. introduced the first commercially available implementation of SQL, Oracle V2 (Version2) for VAX computers. *Oracle V2* beat IBM's release of the System/38 RDBMS to market by a few weeks.

After testing SQL at customer test sites to determine the usefulness and practicality of the system, IBM began developing commercial products based on their System R prototype including System/38, SQL/DS, and DB2, which were commercially available in 1979, 1981, and 1983, respectively.

Standardization

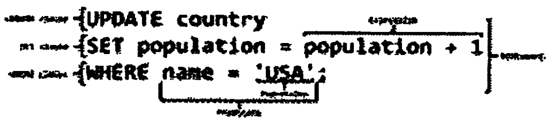
SQL was adopted as a standard by ANSI in 1986 and ISO in 1987. In the original SQL standard, ANSI declared that the official pronunciation for SQL is "es queue el".

Until 1996, the National Institute of Standards and Technology (NIST) data management standards program was tasked with certifying SQL DBMS compliance with the SQL standard. In 1996, however, the NIST data management standards program was dissolved, and vendors are now relied upon to self-certify their products for compliance.

The SQL standard has gone through a number of revisions, as shown below:

Year	Name	Alias	Comments
1986	SQL-86	SQL-87	First published by ANSI. Ratified by ISO in 1987.
1989	SQL-89	FIPS 127-1	Minor revision, adopted as FIPS 127-1.
1992	SQL-92	SQL2, FIPS 127-2	Major revision (ISO 9075), Entry Level SQL-92 adopted as FIPS 127-2.
1999	SQL:1999	SQL3	Added regular expression matching, recursive queries, triggers, support for procedural and control-of-flow statements, non-scalar types, and some object-oriented features.
2003	SQL:2003		Introduced XML-related features, window functions, standardized sequences, and columns with auto-generated values (including identity-columns).
2006	SQL:2006		ISO/IEC 9075-14:2006 defines ways in which SQL can be used in conjunction with XML. It defines ways of importing and storing XML data in an SQL database, manipulating it within the database and publishing both XML and conventional SQL-data in XML form. In addition, it provides facilities that permit applications to integrate into their SQL code the use of XQuery, the XML Query Language published by the World Wide Web Consortium (W3C), to concurrently access ordinary SQL-data and XML documents.

Language elements



This chart shows several of the SQL language elements that compose a single statement.

The SQL language is sub-divided into several language elements, including:

- Statements which may have a persistent effect on schemas and data, or which may control transactions, program flow, connections, sessions, or diagnostics.
- Queries which retrieve data based on specific criteria.
- Expressions which can produce either scalar values or tables consisting of columns and rows of data.
- Predicates which specify conditions that can be evaluated to SQL three-valued logic (3VL) Boolean truth values and which are used to limit the effects of statements and queries, or to change program flow.
- Clauses which are (in some cases optional) constituent components of statements and queries.
- Whitespace is generally ignored in SQL statements and queries, making it easier to format SQL code for readability.
- SQL statements also include the semicolon (";") statement terminator. Though not required on every platform, it is defined as a standard part of the SQL grammar.

Queries

The most common operation in SQL databases is the query, which is performed with the declarative SELECT keyword. SELECT retrieves data from a specified table, or multiple related tables, in a database. While often grouped with Data Manipulation Language (DML) statements, the standard SELECT query is considered separate from SQL DML, as it has no persistent effects on the data stored in a database. Note that there are some platform-specific variations of SELECT that can persist their effects in a database, such as the SELECT INTO syntax that exists in some databases.

SQL queries allow the user to specify a description of the desired result set, but it is left to the devices of the database management system (DBMS) to plan, optimize, and perform the physical operations necessary to produce that result set in as efficient a manner as possible. An SQL query includes a list of columns to be included in the final result immediately following the SELECT keyword. An asterisk ("*") can also be used as a "wildcard" indicator to specify that all available columns of a table (or multiple tables) are to be returned. SELECT is the most complex statement in SQL, with several optional keywords and clauses, including:

- The FROM clause which indicates the source table or tables from which the data is to be retrieved. The FROM clause can include optional JOIN clauses to join related tables to one another based on user-specified criteria.
- The WHERE clause includes a comparison predicate, which is used to restrict the number of rows returned by the query. The WHERE clause is applied before the GROUP BY clause. The WHERE clause eliminates all rows from the result set where the comparison predicate does not evaluate to True.
- The GROUP BY clause is used to combine, or group, rows with related values into elements of a smaller set of rows. GROUP BY is often used in conjunction with SQL aggregate functions or to eliminate duplicate rows from a result set.
- The HAVING clause includes a comparison predicate used to eliminate rows after the GROUP BY clause is applied to the result set. Because it acts on the results of the GROUP BY clause, aggregate functions can be used in the HAVING clause predicate.
- The ORDER BY clause is used to identify which columns are used to sort the resulting data, and in which order they should be sorted (options are ascending or descending). The order of rows returned by an SQL query is never guaranteed unless an ORDER BY clause is specified.

Data manipulation

First, there are the standard Data Manipulation Language (DML) elements. DML is the subset of the language used to add, update and delete data:

- INSERT is used to add rows (formally tuples) to an existing table, for example:

INSERT INTO My_table (field1, field2, field3) VALUES ('test', 'N', NULL);

- UPDATE is used to modify the values of a set of existing table rows, eg:

UPDATE My_table SET field1 = 'updated value' WHERE field2 = 'N';

- DELETE removes zero or more existing rows from a table, eg:

DELETE FROM My_table WHERE field2 = 'N';

- MERGE is used to combine the data of multiple tables. It is something of a combination of the INSERT and UPDATE elements. It is defined in the SQL:2003 standard; prior to that, some databases provided similar functionality via different syntax, sometimes called an "upsert".

Transaction controls

Transactions, if available, can be used to wrap around the DML operations:

- START TRANSACTION (or BEGIN WORK, or BEGIN TRANSACTION, depending on SQL dialect) can be used to mark the start of a database transaction, which either completes entirely or not at all.
- COMMIT causes all data changes in a transaction to be made permanent.

- ROLLBACK causes all data changes since the last COMMIT or ROLLBACK to be discarded, so that the state of the data is "rolled back" to the way it was prior to those changes being requested.

Once the COMMIT statement has been executed, the changes *cannot* be rolled back. In other words, its meaningless to have ROLLBACK executed after COMMIT statement and vice versa.

COMMIT and ROLLBACK interact with areas such as transaction control and locking. Strictly, both terminate any open transaction and release any locks held on data. In the absence of a START TRANSACTION or similar statement, the semantics of SQL are implementation-dependent.

Example:

A classic bank transfer of funds transaction.

```
START TRANSACTION;
  UPDATE Account SET amount=amount-200 WHERE account number=1234;
  UPDATE Account SET amount=amount+200 WHERE account number=2345;
IF ERRORS=0 COMMIT;
IF ERRORS<>0 ROLLBACK;
```

Data definition

The second group of keywords is the Data Definition Language (DDL). DDL allows the user to define new tables and associated elements. Most commercial SQL databases have proprietary extensions in their DDL, which allow control over nonstandard features of the database system. The most basic items of DDL are the CREATE, ALTER, RENAME, TRUNCATE and DROP statements:

- CREATE causes an object (a table, for example) to be created within the database.
- DROP causes an existing object within the database to be deleted, usually irretrievably.
- TRUNCATE deletes all data from a table (non-standard, but common SQL statement).
- ALTER statement permits the user to modify an existing object in various ways -- for example, adding a column to an existing table.

Example:

```
CREATE TABLE My_table (
  my_field1  INT,
  my_field2  VARCHAR (50),
  my_field3  DATE         NOT NULL,
  PRIMARY KEY (my_field1, my_field2)
```

Data control

The third group of SQL keywords is the Data Control Language (DCL). DCL handles the authorization aspects of data and permits the user to control who has access to see or manipulate data within the database. Its two main keywords are:

- GRANT authorizes one or more users to perform an operation or a set of operations on an object.
- REVOKE removes or restricts the capability of a user to perform an operation or a set of operations.

Example:

GRANT SELECT, UPDATE ON My_table TO some_user, another_user;

SQl is the base language for databases. For getting maximum benefits of database designing, some database tools are used in modern world.

5.2.3 JAVA IDEs

In the modern competitive world Time has become everything with that reason creating software has become a great challenge to overcome that challenge JAVA programmer considered the best and fast ways to present Quality software ,that system named as Integrated Development Environments or in short form IDE. Integrated Development Environments (IDE) provides benefits to programmers that plain text editors cannot match. IDEs can parse source code as it is typed, giving it a syntactic understanding of the code. This allows advanced features like code generators, auto-completion, refectory and debuggers.

IDEs are programs designed to make programming easier

- Graphical Interface is intuitive
- Program management is easier
- Source code readability is improved
- Too many features to list

You do not need an IDE to create a program. But they are very helpful.

Present use Java IDEs in brief

- BlueJ
- CodeLab
- Code Warrior
- DrJava
- Eclipse
- IDLE
- JBuilder
- JCreator
- JES
- jGrasp
- Jython
- NetBeans
- Python
- Sun's JDK/SDK
- TextPad

JBuilder

JBuilder Professional is a comprehensive, visual development environment for creating JAVA 2, platform-independent database and Web applications. JBuilder combines a rapid application development environment for building and deploying JavaBeans, applets, servelets, and Java/XML. This environment includes extensive source management and a professional graphical debugger.

JCreator

JCreator is a powerful IDE for Java. JCreator provides the user with a wide range of functionality such as : Project management, project templates, code-completion, debugger interface, editor with syntax highlighting, wizards and a fully customizable user interface. With JCreator you can directly compile or run your Java program without activating the main document first. JCreator will automatically find the file with the main method or the html file holding the java applet, then it will start the tool.

NetBeans

The original free and open source IDE. Develop cross-platform desktop, mobile and web applications based on industry standards utilizing the latest technologies with full-featured integrated development environment for Java Software Developers.

Python

Python is an interpreted, interactive, object-oriented programming language. It is often compared to Tcl, Perl, Scheme or Java. Python combines remarkable power with very clear syntax. It has modules, classes, exceptions, very high level dynamic data types, and dynamic typing. There are interfaces to many system calls and libraries, as well as to various windowing systems (X11, Motif, Tk, Mac, MFC). New built-in modules are easily written in C or C++. Python is also usable as an extension language for applications that need a programmable interface. The Python implementation is portable: it runs on many brands of UNIX, on Windows, OS/2, Mac, Amiga, and many other platforms. The Python implementation is copyrighted but freely usable and distributable, even for commercial use.

The Main Features of MySQL:

Internals and Portability

Written in C and C++.

Uses GNU Automake (1.4), Autoconf (Version 2.52 or newer), and Libtool for portability.

Works on many different platforms; APIs for C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, and Tcl.

Fully multi-threaded using kernel threads, this means it can easily use multiple CPUs if available.

Very fast B-tree disk tables with index compression and thread-based memory allocation system.

Very fast joins using an optimised one-sweep multi-join, can mix tables from different databases in the same query.

In-memory hash tables which are used as temporary tables.

SQL functions are implemented through a highly optimised class library and should be as fast as possible!

Usually there isn't any memory allocation at all after query initialisation.

Security

All password traffic is encrypted connecting to a server.

A privilege and password system that is very flexible and allows host-based verification.

Scalability and Limits

Handles large databases. Maximum size for a table is 8TB (default 4GB).

Up to 32 indexes per table. Each index may consist of 1 to 16 columns or parts of columns.

The maximum index width is 500 bytes (this may be changed when compiling MySQL Server).

An index may use a prefix of a CHAR or VARCHAR field.

Connectivity

Clients may connect to the MySQL server using TCP/IP Sockets, Unix Sockets (Unix), or Named Pipes (NT).

All ODBC 2.5 functions and many others.

Localization

All comparisons for normal string columns are case-insensitive.

The server can provide error messages to clients in many languages.

MySQL Server supports many different character sets that can be specified at compile and runtime.

Clients and Tools

Includes myisamchk, a very fast utility for table checking, optimisation, and repair.

All of the functionality of myisamchk is also available through the SQL interface as well.

All MySQL programs can be invoked with the --help or -? options to obtain online assistance.

Growth of MySQL

MySQL represents the most impressive market success, exceeded only perhaps by Apache in free and open source software. In terms of installed base, MySQL has left the technically impressive rival
PostgreSQL in the dust. It has marginalized mSQL, SQLite, and SAP DB

It has started to challenge the proprietary database companies on their own turf, asalready mentioned. Nobody
can say why licensing costs for proprietary databases haveplummeted in recent years, but one suspects that it's due to
MySQL's competition, as are the large discounts Microsoft has offered certain customers.

Charmed status of the MySQL

A textbook case of a disruptive technology

MySQL, first of all, illustrates in almost pure form thesequence of events Clayton M. Christensen documented as a "disruptive technology" in his ground-breaking book *The Innovator's Dilemma*. Early versions of MySQL lacked thebasic features, such as ACID transactions and referentialintegrity, that experienced users expected from a relational database. In a pattern familiar to anyone who has read Christensen's book, knowledgeable observers dismissed MySQL as a toy.

But MySQL's very simplicity made it so small and fast that it quickly won over small users who wouldn't even understand what they were missing and how to use the fancy features offered by "real" database engines. In particular, MySQL proved ideal for the exploding area of dynamic Web content.

Most indicative of its mantle as a true disruptive technology, MySQL proved that many of the missing high-end features weren't as indispensable as people used to claim. For instance, referential integrity (jeez, who could be opposed to integrity?) wasn't required in a database when it could be achieved in the application code, often more reliably. You could also achieve efficient locking without row-level locks; in fact, supporting row-level locks took so much overhead that the application was almost better without them.

Having rewritten the rules for what constituted a useful relational database engine, MySQL AB proceeded to invest resources to implement the very features which they were originally sneered at for lacking. Bit by bit they have added check-off items to their T-shirts. And what's most interesting is how they found the resources to pull off this kind of upgrade cycle.

The importance of dual-licensing

Of course, any agreement under which you release free software (other than the public domain) is a license, but "licensing" usually refers to selling licenses. And MySQL AB has become one of most successful companies with a completely complementary dual-licensing model: they offer everything under an open license for certain users, but charge money for everything under other circumstances.

As we'll see, the parallel existence of GPL licensing and commercial licensing leaves a mark on every aspect of the company.

The CEO of MySQL AB, Marten Mickos, said that more than half of their money comes from license fees. This contrasts with an impression of open source software left by Novell vice president Chris Stone in his keynote.

Stone, claiming that Novell had already settled on a maintenance model for revenue, suggested that, because of this, the move to open source will not be as hard for Novell as for other traditional computer companies. The remarks implied that an open source business model has to be a support model, but MySQL AB staff pointed out that support contracts have been shown to be insufficient to fund software development. It may be enough in the future, but it's not yet.

The other side of dual licensing is equally important. In terms of adoption, open licenses do more for a software project than twenty thousand billboards and glossy ads. The GPL allowed MySQL to penetrate millions of sites that would never have otherwise known about it.

MySQL AB also benefits directly from contributions; for instance, its most feature-rich storage engine, InnoDB, started as an outside project.

But MySQL would have remained a stepping stone to other databases for many people, were it not for its continual growth and improvement. This rate of improvement is not exceedingly fast (managers stress that they always check for stability, correctness, and performance before releasing enhancements) but it's fast enough to give customers the impression that features are worth waiting for–that what they want will in due time be added to the product.

And there's a symbiosis between technical development and payments for licenses. Each requires the other. If a substantial body of enhancements to MySQL grew up outside the company–even if they were put under the GPL and MySQL AB could incorporate them into its version of MySQL–they would not be part of the value MySQL AB could offer paying customers. There would thus be few paying customers, and MySQL AB could not afford to hire people to keep up development. In order to keep up with customer needs, MySQL AB has managed one of the coolest tricks in open source development: keeping most development in-house. And making users happy about it!

Whereas Linux and Apache belong to everybody and nobody, MySQL is taken seriously by large companies with money to spend because there's a company that owns a trademark on it and markets it like a proprietary product.

So MySQL succeeds at maintaining two faces. To paying customers, it's a traditional, responsible vendor. To programmers and database administrators, it's a flexible, responsive network of independently-minded developers in free-software style.

SAP adds its muscle

Nobody would be sorry to have the backing that comes from such a large and well-established corporation as SAP. But in addition to SAP's prestige and endorsement of MySQL, what is the main contribution of the partnership?

Not MaxDB. This is the new name for SAP DB, and was honored with several sessions at the conference, all poorly attended.

And probably not the money SAP invested in MySQL AB as part of the partnership they announced in May 2003. Certainly this helped to spur the enormous hiring campaign MySQL has been on during the past year.

In particular, MySQL 5.1 is supposed to contain server-side cursors, views, standard error handling, standard security handling, schemas, and constraints.

There are three reasons for incorporating SAP DB features into MySQL:

1. They are genuinely useful.
2. They are needed to run SAP.
3. They are ANSI-compliant.

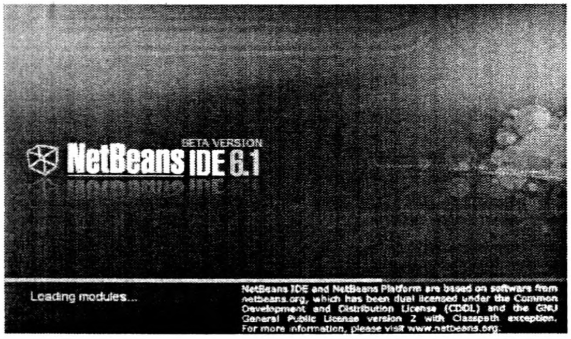
Java Database Connectivity

Java Database Connectivity (JDBC) is an API for the Java programming language that defines how a client may access a database. It provides methods for querying and updating data in a database. JDBC is oriented towards relational databases

The selected Java IDE for FISHER 2009

After, the latest IDEs have been considered we decided to choose Net Beans as our FISHER2009 Software due to its' user friendly features and it is totally free but the key point was it the IDE that SUN Microsystems has recommended.FISHER2009 was started with Net Beans 5.5 but when the process was going Net Beans 6.0 and Net Beans 6.1 has been implemented.

NETBEANS 6.1



Loading modules...

NetBeans IDE and NetBeans Platform are based on software from netbeans.org, which has been dual licensed under the Common Development and Distribution License (CDDL) and the GNU General Public License version 2 with Classpath exception. For more information, please visit www.netbeans.org.

NetBeans IDE 6.1 Information

The NetBeans IDE is a modular, standards-based, integrated development environment (IDE) written in the Java programming language. The NetBeans project consists of an open source IDE and an application platform, which can be used as a generic framework to build any kind of application.

Release Overview

The NetBeans IDE 6.1 release provides several new features and enhancements, such as rich JavaScript editing features, support for using the Spring web framework, tighter MySQL integration, and an improved way of sharing libraries among dependent projects. The acclaimed support for Ruby/JRuby has been enhanced with new editor quick fixes, a Ruby platform manager, fast debug support for JRuby, and many other new features and fixes.

By popular demand, the bean pattern and JSF CRUD generation features that were missing in the 6.0 release have returned. In addition, early versions of new modules, such as ClearCase support, are available as plugins.

This release also provides improved performance, especially faster startup (up to 40%), lower memory consumption and improved responsiveness while working with large projects. See below for a list of features in this release.

NetBeans IDE 6.1 – Faster, Better, Stronger

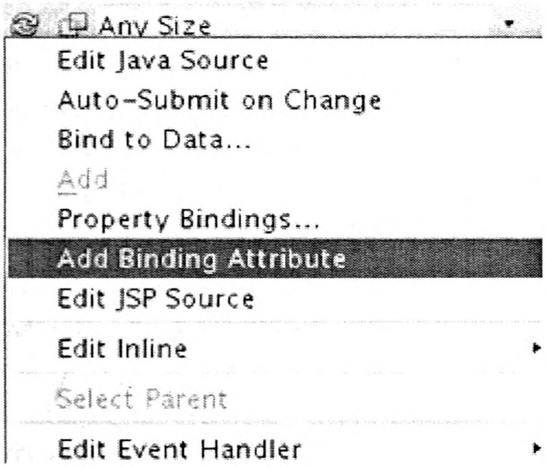
NetBeans IDE version 6.1 has been released recently. This version comes relatively soon after its widely successful predecessor – NetBeans IDE 6.0. Although NetBeans 6.1 is not as revolutionary as 6.0 (which brought a completely redesigned Java editor), the new release has many new features. In this article we'll discuss the new features one by one. This article only covers the major improvements, so if you want to see a complete list please visit the 6.1 New and Noteworthy page.

Let's look at what is new and improved in the new release.

Performance and Quality

The main themes of the release are performance and quality – after all, as a minor release it stabilizes the previous major release. These goals are rather intangible but developers should notice a faster startup (up to 40% over version 6.0 if multiple projects are open) and different performance boosts all across the board. One of the big issues in 6.0 was slow parsing of JSP files, and feedback from the NetBeans community indicates that version 6.1 doesn't suffer from this problem anymore. A new incremental parser has been integrated into the Java editor, so all Java syntax related features such as code completion, navigator, refactorings, etc. should be noticeably faster, especially on large classes. Several I/O related optimizations have been used to reduce the number of disk accesses, improving responsiveness in many cases.

One performance improvement needs closer examination – the Visual Web Designer received many performance-related fixes leading to lower memory usage. The performance team fixed several issues with memory leaks which may have caused the Visual Designer to grow consumption of memory over time. The most significant change, though, is that the binding attributes no longer get generated by default – which leads to many performance improvements because the classes generated by Visual Web are much smaller and do not include unnecessary attributes, getters and setters. However, this change may be surprising to some users – you need to make sure you generate the necessary binding attribute to have access to the element you want to manipulate, as seen on the screenshot:



In the beginning it may be surprising that you need to add binding attributes for each page element you want to manipulate, but the performance gains are definitely worth the extra work.

New JavaScript Editor

Screencast: New JavaScript editor in NetBeans 6.1

NetBeans 6.1 provides a brand new JavaScript editor based on the GSF framework (General

Scripting Framework) which was introduced together with Ruby editor in 6.0. It took only a few months to provide many new JavaScript editing features, such as:

- Semantic highlighting
- Mark occurrences
- Instant rename
- Rename refactoring
- Quick fixes and semantic checks
- Tasklist integration
- Code completion and type analysis
- JavaScript documentation in code completion
- Browser compatibility information in code completion
- Go to declaration
- Open JavaScript type
- ... and much more.

The editing experience with JavaScript in version 6.1 is similar to the Java and Ruby editors. Work on a JavaScript debugger is in progress and its first prototype should be demo-ed at NetBeans Day in San Francisco in May 2008.

MySQL Support

Due to the recent acquisition of MySQL AB by Sun Microsystems, the NetBeans IDE 6.1 added integration with MySQL. You can start or stop the MySQL server right from the IDE. A default connection is generated for you, and you can browse database tables easily and create connections to these tables with one click. Getting started developing with the NetBeans IDE and MySQL is even easier than before.

Mobility

In the Java ME area, Mac OS X is now officially supported and the Mpower emulater can be easily used from the IDE. Several new SVG components have been added and the quality and stability of Mobility Pack has been enhanced as well.

RESTful Web Services Support

The RESTful web service support in the Netbeans IDE is based on the JSR 311 standard. The IDE has a wizard to create RESTful services from JPA entity classes. You can also create RESTful services based on popular design patterns provided by the IDE. Another wizard generates JavaScript client stubs that invoke these services. A popular feature is the test client that provides an interactive way to test and view the result of web service invocations. RESTful web service support was available since NetBeans 6.0 as a plugin, now it is part of the Netbeans IDE 6.1 standard distribution.

6.0 exe4j

exe4j is a Java exe maker that helps you integrate your Java applications into the Windows operating environment, whether they are service, GUI or command line applications. If you want your own process name instead of java.exe in the task manager and a user friendly task-bar grouping in Windows XP, exe4j does the job. exe4j helps you with starting your Java applications in a safe way, displaying native splash screens, detecting or distributing suitable JREs and JDKs, startup error handling and much more.

The use of exe4j s use is not time limited, but restricted to evaluation purposes. Evaluation warnings

are removed after purchasing an exe4j license .

exe4j Licensing

Without a valid license, exe4j may be used for evaluation purposes only. The evaluation period is not

time limited, but excludes any use for creating distributions of commercial products.

exe4j licenses can be purchased easily and securely online. They accept a large variety of payment

methods including credit cards, checks and purchase orders. Pricing information is available online.

The license key can enter in the welcome step of the exe4j wizard. If a license has been

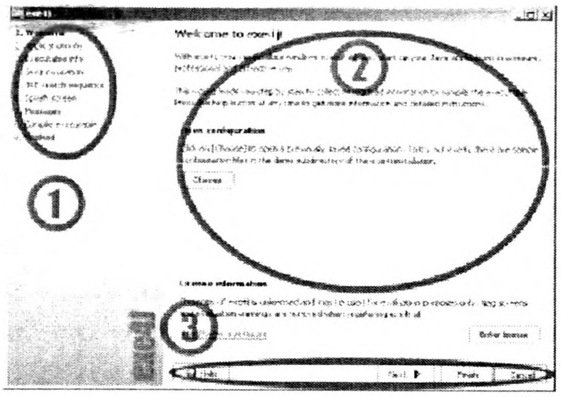
entered, the licensing information is visible there. The exe4j command line compiler also prints

licensing information except when invoked with the quiet option.

exe4j wizard

When invoking exe4j from the start menu, the desktop icon or by executing *bin\exe4j.exe* in the exe4j installation directory, the exe4j wizard is started. It guides you step by step through completing the required information for building the executable.

The window of the wizard has three distinct areas:



(1) Index

The index of steps lists all steps of the wizard and highlights the current step in bold face. You can click on any step in the index to arbitrarily move between wizard steps. Alternatively, you can use the navigation controls.

- (2) Current step

The information for the current step is entered here. See the help pages for each step for specific information.

- (3) Navigation

The navigation bar allows you to move back and forward through the steps of the wizard with the **[Next]** and **[Previous]** buttons. The **[Finish]** button allows you to complete the wizard immediately without moving through the remaining steps. Should any required information be missing, the wizard will alert you to it. The **[Help]** button shows context specific help that is also available by pressing **F1**. With the **[Cancel]** button you can leave the wizard at any time. Should you have made modifications to the configuration, you will be asked whether you want to save your changes before the wizard exits.

By default, the wizard starts with a template config file that contains default values where appropriate.

The template config file is loaded from *config\template.exe4j* in the exe4j installation directory.

If you would like to load a config file at startup, you can pass the path of the desired config file to the wizard executable (*bin\exe4j.exe* in the exe4j installation directory).

Version Info

A version info resource will enable the Windows operating system to determine meta information about your executable. This information is displayed in various locations. For example, when opening the property dialog for the executable in the Windows explorer, a "Version" tab will be present in the property dialog if you have chosen to generate the version info resource. The version info resource consists of several pieces of information. If you check Generate version info resource, there are several fields whose values must be entered in the text fields on this step. Note that the "original file name" and the "product name" fields in the version info resource are filled in automatically by exe4j.

- Product version

The product version must be composed of a maximum of 4 numbers, separated by spaces, commas or dots. By using the -r flag for the command line compiler, the product version can be overidden. The product version is also used in the splash screen version line config as a replacement value for the variable %VERSION%.

- File version

If you want to specify a version for the file which is a different from the product version, you can do it here. If this field is left empty, the product version will be used for the file version.

- Internal name

Choose a short internal name for identifying your application.

- Company name

Enter the name of your company.

- File description

Enter a description of the application.

- Legal copyright

Enter a copyright statement for your application.

Native library directories

If your application uses native libraries that you would lke to load with a System.loadLibrary() call, the directory where the *.dll* is located must be included in the **PATH** environment variable.

You can add such directories in the path list of this step.

- Add native library directory (key INS)

Lets you add a new directory to the end of the list. Choose the native library directory in the file chooser that appears after clicking this button. The directory will be converted to a path relative to the distribution source directory.

- Remove native library directory (key DEL)

Removes the currently selected native library directory entry without further confirmation.

- Move entry up (key ALT-UP)

Moves the selected native library directory entry up one position in the path list.

- Move entry down (key ALT-DOWN)

Moves the selected native library directory entry down one position in the path list.

Configure Search Sequence

The search sequence list shows all search sequence entries that have been added so far. For new configurations, a default search sequence is pre-defined.

The following types of search sequence

entries are available:

- Search registry
- Directory
- Environment variable

The control buttons on the right allow you to modify the contents of the search sequence list:

- Add search sequence entry (key INS)

Invokes the search sequence entry dialog. Upon closing the search sequence entry dialog with the **[OK]** button, a new search sequence entry will be appended to the bottom of the search

sequence list.

- Remove search sequence entry (key DEL)

Removes the currently selected search sequence entry without further confirmation.

- Move search sequence entry up (key ALT-UP)

Moves the selected search sequence entry up one position in the class path list.

- Move search sequence entry down (key ALT-DOWN)

Moves the selected search sequence entry down one position in the class path list.

It is possible to generate a log file that contains information about the JRE search sequence and any potential problems. In order to switch on logging, please define the environment variable EXE4J_LOG=yes and look for the newest text file whose name starts with i4j_nlog_ in the Windows temp directory. This information can be used for debugging purposes.

Preferred VM

- default VM

exe4j will use the default VM for the found JRE.

- client hotspot VM

exe4j will try to use the client hotspot VM for the found JRE. This is equivalent to using the –client switch when invoking java from the command line.

- server hotspot VM

exe4j will try to use the server hotspot VM for the found JRE. This is equivalent to using the –server switch when invoking java from the command line. Please note that it is not an error if the selected JVM is not present for the found JRE. exe4j will simply use another JVM to launch your application in that case.

Splash Screen Options

The behavior of the splash screen can be defined in the General section of this step:

- **Hide splash screen when first application window is shown**

If this option is checked, the exe4j executable will monitor the state of your application and hide the native splash screen as soon as a window is opened. If you want to hide the splash screen programmatically, you can use exe4j's launcher API.

- **Splash screen is always on top**

If this option is checked, the splash screen remains always on top of other windows opened by your application.

The Status line and Version line sections allow you to position the text lines on the splash screen and configure their font. The status line is dynamically updatable with exe4j's launcher API. If you include the variable %VERSION% in the version line text, it will be replaced with the product version defined in the version info step of the wizard. With the -r flag, you can override this setting for the command line compiler .

You can configure the following properties of a text line

- Text

The (initial) text displayed in the text line.

- Position

The x and y-coordinates of the text line on the splash screen. The origin of the coordinate system is the top left corner of the splash screen window.

- Font

The font used for drawing the text line:

- Name

The name of the font. Please choose a common font name that is likely to be available on all target platforms. If unavailable at runtime, the MS Dialog font will be used as a fallback.

- Weight

The weight of the font. The 8 Windows standard font weights are offered as a choice.

- Size

The size of the font in points.

- Color

The color of the font. By clicking on **[...]**, a color chooser dialog is brought up.

To **visually position the text lines** with mouse and keyboard on the actual splash screen image,please click on the **[Position text lines visually]** button. The visual positioning dialog will then be displayed. On exiting the dialog with the **[OK]** button, the X/Y coordinate text fields will be updated for both text lines.

exe4j features

- Customized JRE/JDK detection

The executable can detect appropriate Java JREs and JDKs in the Windows registry, in environment variables, special directories and on the system path. You can fully customize the search sequence, error handling and supported JRE/JDK versions.

- Optional distribution of a bundled JRE

exe4j allows you to distribute your own private JRE with your application. This way you can ensure that your application's requirements are definitely met. You can even configure where the JRE is located.

- Flexible classpath construction

The classpath for your Java application can be fully customized to scan directories for JAR files, include specific directories and archives as well as insert environment variables. Customizable error handling allows you to interrupt the startup sequence with a specific error message instead of obscure NoClassDefFound exceptions later on.

- GUI or console applications

exe4j lets you compile GUI applications or console applications with an associated terminal window.

- Windows services

exe4j enables you to easily create a Windows service with Java. With the command line switches /install, /uninstall, /start and /stop you have full control over your service.

- Optional inclusion of JAR files into the executable exe4j lets you include JAR files into the executable - in this way you can distribute your Java applications as a single EXE file.

- Custom process name instead of java/javaw exe4j launches your Java application in such a way, that the exe4j executable and not java.exe or javaw.exe will appear in the task manager. In Windows XP, the task bar grouping will display the name of your executable and the associated icon, instead of the non-descript terminal icon and the string "javaw".

- Custom icon for your executable

exe4j lets you specify an icon file that will be compiled into your executable. This gives your application a much more professional appearance than a batch file or an executable JAR file would.

- Custom working directory

If required you can adjust the working directory to a specific directory relative to the executable. This is especially helpful for console applications which can be invoked from arbitrary directories. This way, you don't need to define fragile environment variables like MYAPP_HOME.

- VM parameters file

For every executable, you can create a user editable VM parameters file. If your executable is called hello.exe, the VM parameters file is called hello.exe.vmoptions and each line in it is added as a single VM parameter.

- Version info resource

exe4j can generate a version info resource entry in your executable. This version info is displayed for example in the property dialog of the Windows explorer. If you wish to obtain the "Designed for Windows" logo, this is an important requirement.

- Native splash screen

For GUI applications, a native splash screen gives the users of your application an optimum feedback about application startup within fractions of a second. Textual status information about application startup and version information can be freely placed on the splash screen. From within your Java code, this status information can be easily updated with one simple call. With exe4j's "auto-off" mode activated, the splash screen is hidden, as soon as your application displays a window.

- Redirection of stderr and stdout

Output stream and error stream can be redirected to customized files. This gives you access to valuable information like an exception stack trace on stderr that would otherwise be lost for a Java GUI application.

- Startup failure detection

No more flashing terminal windows and GUI applications that hang without displaying anything. exe4j executables can monitor a stderr output file and display a native dialog with helpful information to inform about startup failures.

- Optional single application instance enforcement and multiple startup notification

If your application must only be started once, an exe4j generated launcher can enforce this condition. Existing application windows will be brought to the top if a user starts the application a second time. The exe4j API allows you to register a listener that reacts to multiple startups and receives the parameters of the command line.

- Fully localizable messages of the executable

All messages of the executable are localizable. This way, the executable can blend into the target locale of your application.

7.0. Analyze

System specification

Efficiency

To feed data for day to day financial transaction, we have used a sound ways with smooth interfaces in FISHER2009. We use the FISHER2009 software to create standard financial statement and it also 100% assured.

User friendliness

our system has user friendly GUI s which can be understood by any user and also it can avoid many errors.

Upgradeability

Can extent the system according to the user's needs. Adding new features extend the flexibility.

Neediness

In our system we have categorize the main sections and b selecting each section the user can put any data that is needed him or her.

Security

It takes a user name and password to logging the system

Error handling

In our software there cannot be any data redundancy errors that means same data cannot be repeated again and again.

User requirement specification

Financial analyzing

In FISHER2009 one of the required section was financial analyzing, in the business field if some company need to survive one of the compatible thing is to get a pleasant idea of the current situation of the company when the financial section which is binned with FISHER2009 give a perfect idea to the manager about the position of the company.

Financial Forecasting

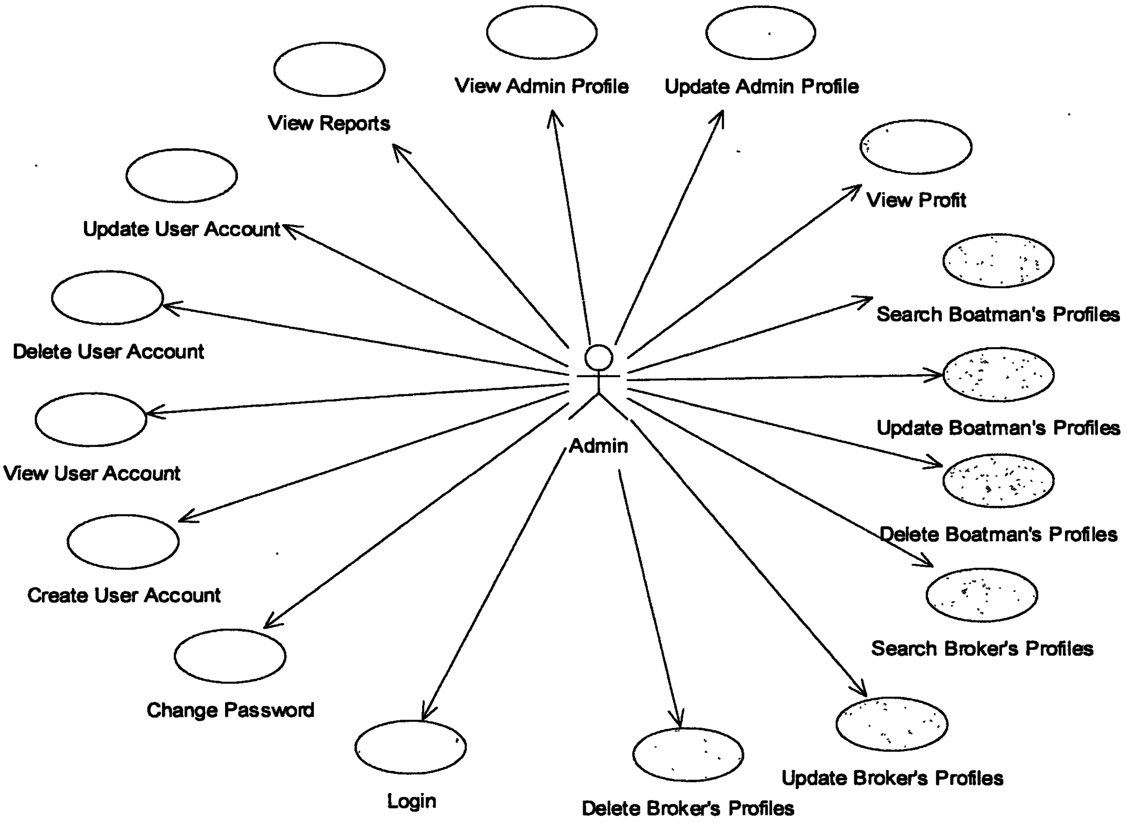
In FISHER2009 one of the most required sections was financial forecasting, in the modern business field. If businessman need to survive and be developed for that the administration must plane the future and must take the decision to fulfill the financial targets.

8.0 Designed

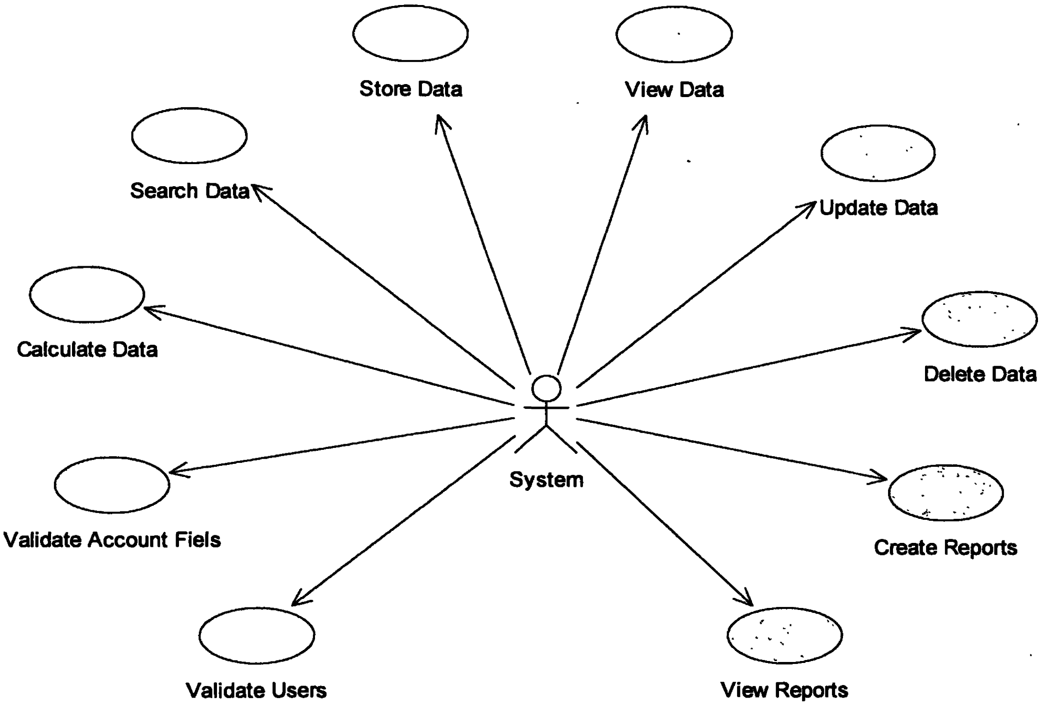
8.1 System designed

8.1.1 Usecase diagrams

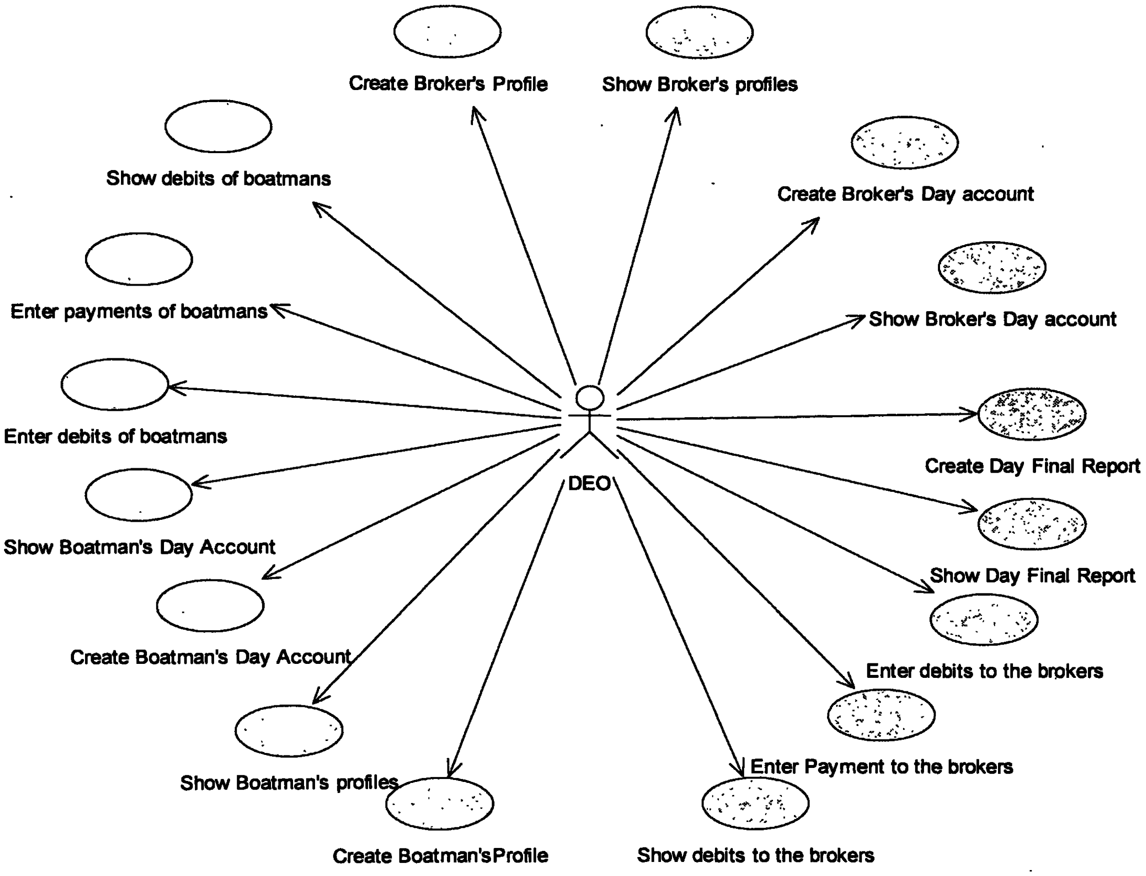
Administrator



System

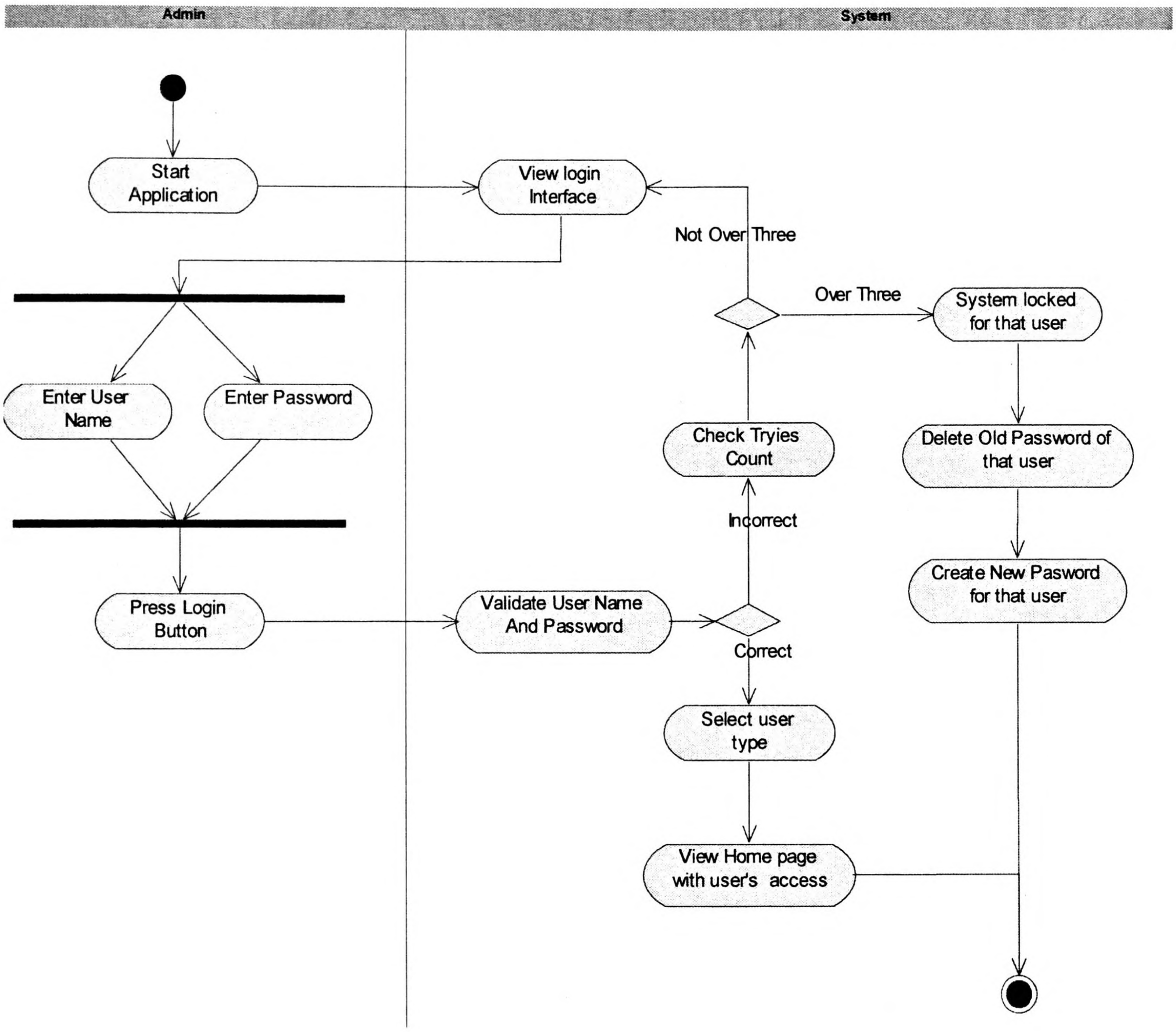

Store Data
View Data
Search Data
Update Data
Calculate Data
Delete Data
System
Validate Account Fiels
Create Reports
Validate Users
View Reports

Data Entry Operator

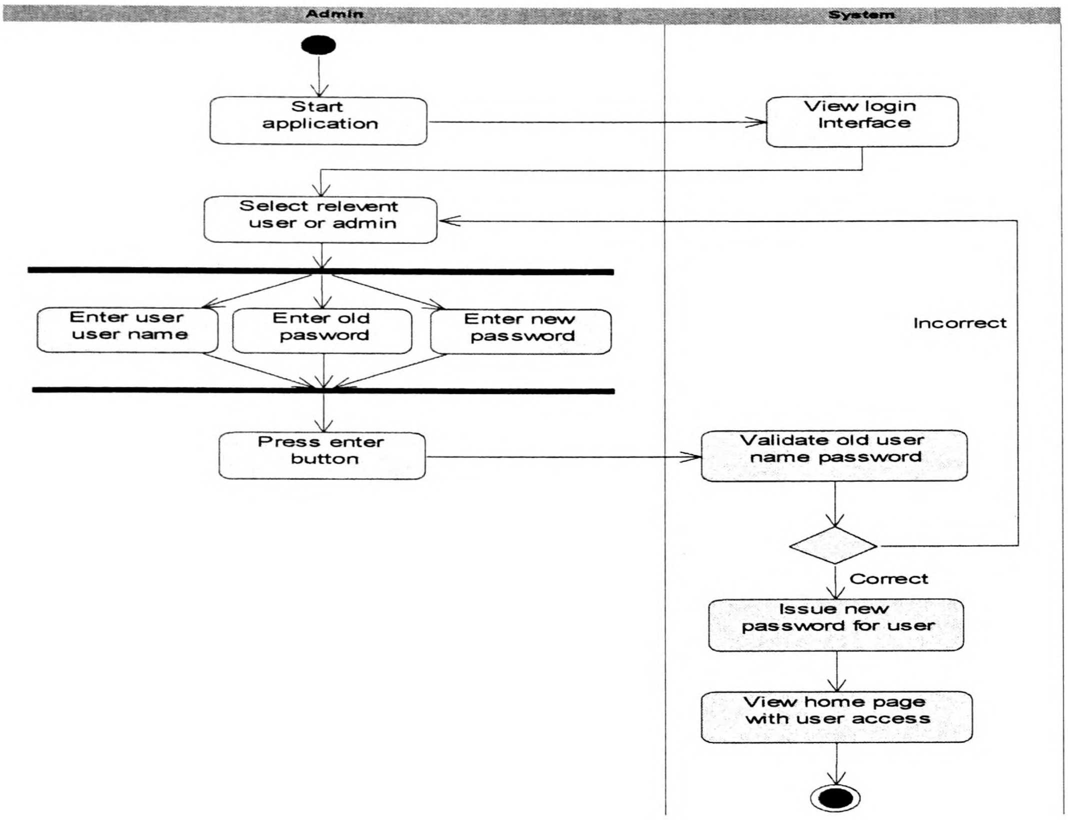

Create Broker's Profile
Show Broker's profiles
Show debits of boatmans
Create Broker's Day account
Enter payments of boatmans
Show Broker's Day account
Enter debits of boatmans
DEO
Create Day Final Report
Show Boatman's Day Account
Show Day Final Report
Create Boatman's Day Account
Enter debits to the brokers
Show Boatman's profiles
Enter Payment to the brokers
Create Boatman'sProfile
Show debits to the brokers

8.1.2 Activity diagrams

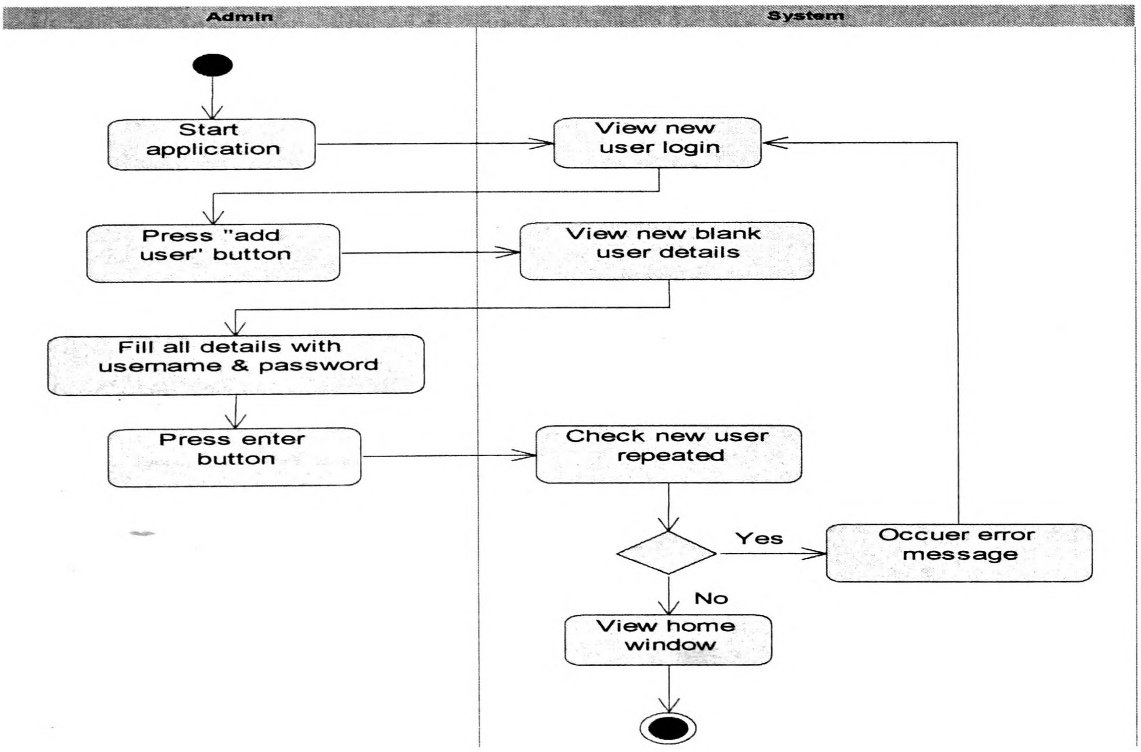
Logging



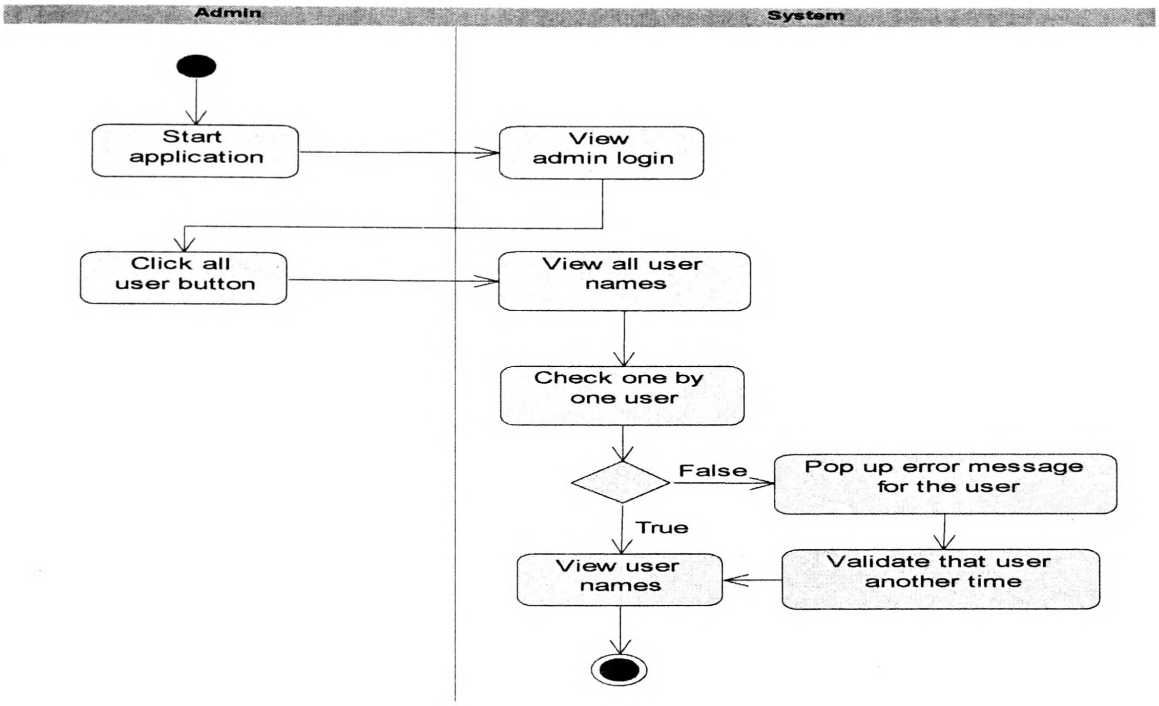
Change Password



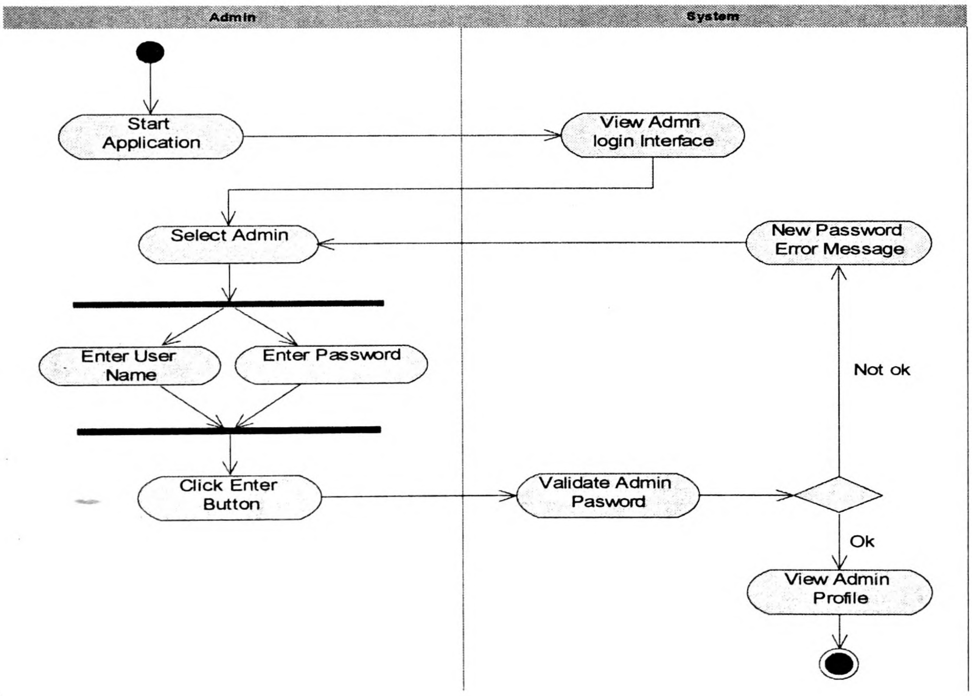
Create User Account



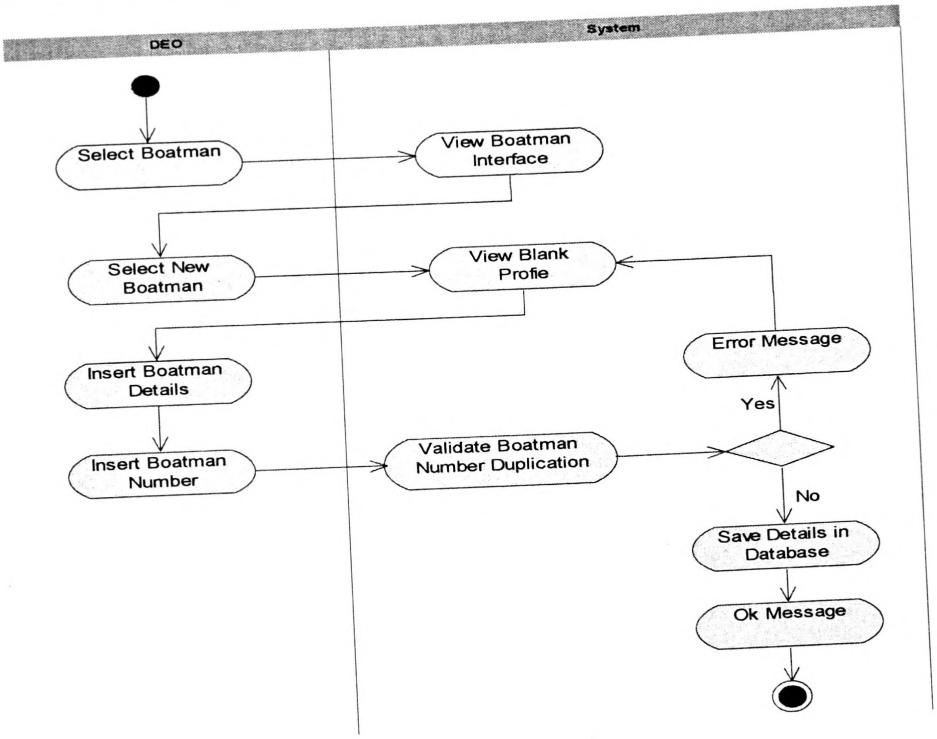
View User Account



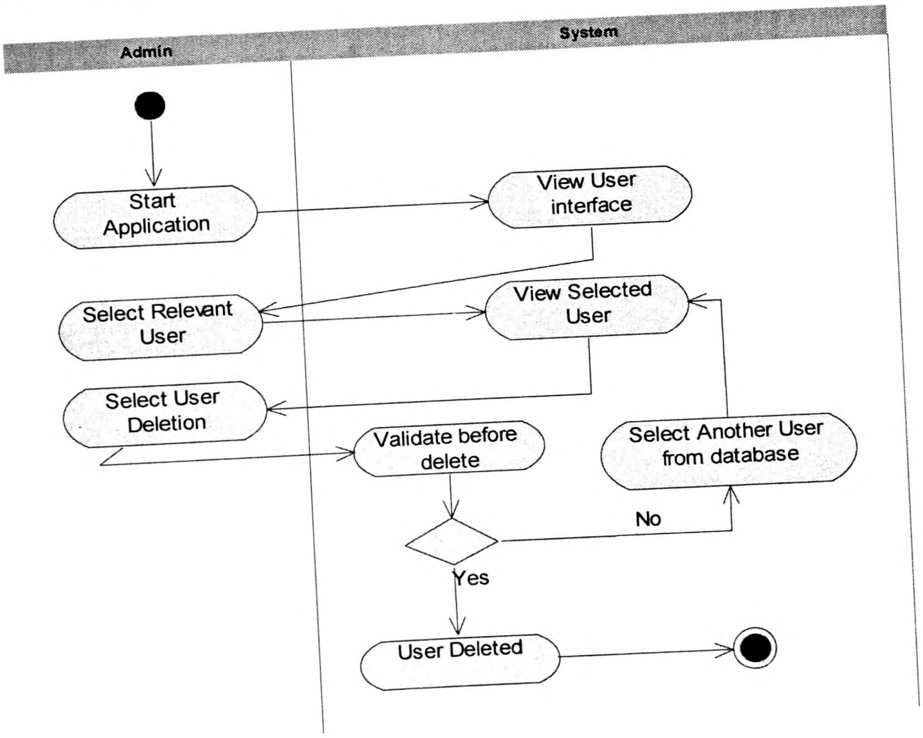
View Admin Profile



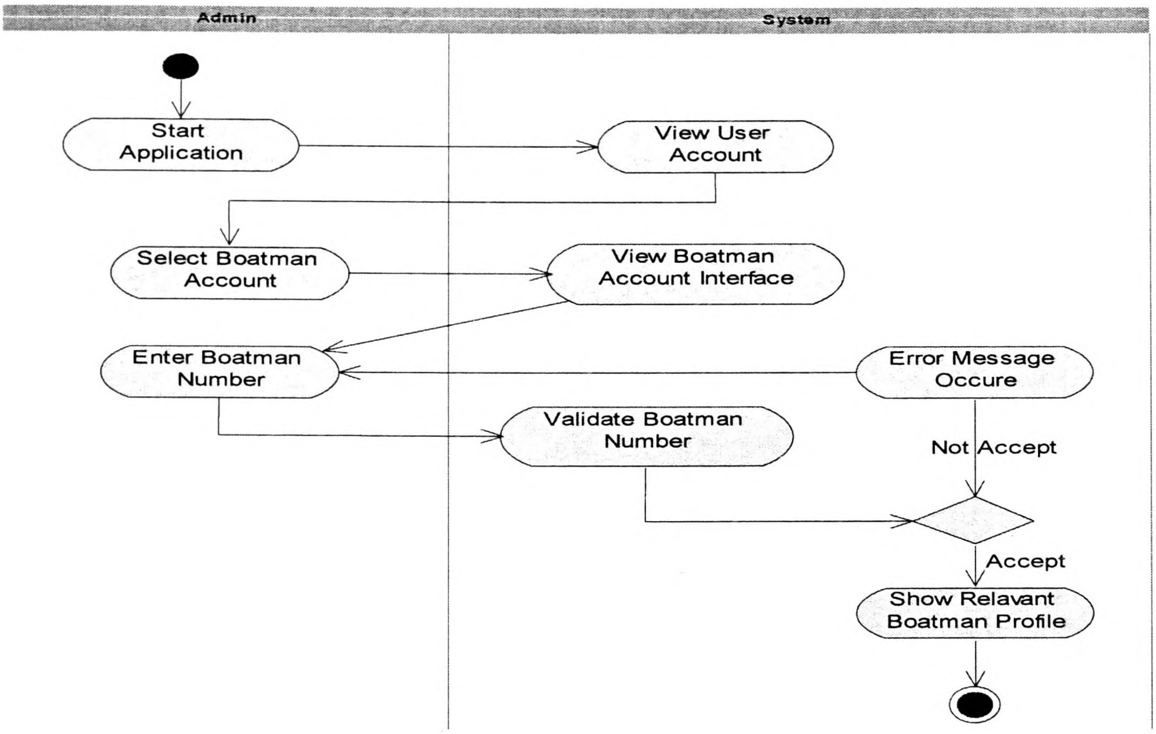
Create Boatman Profile



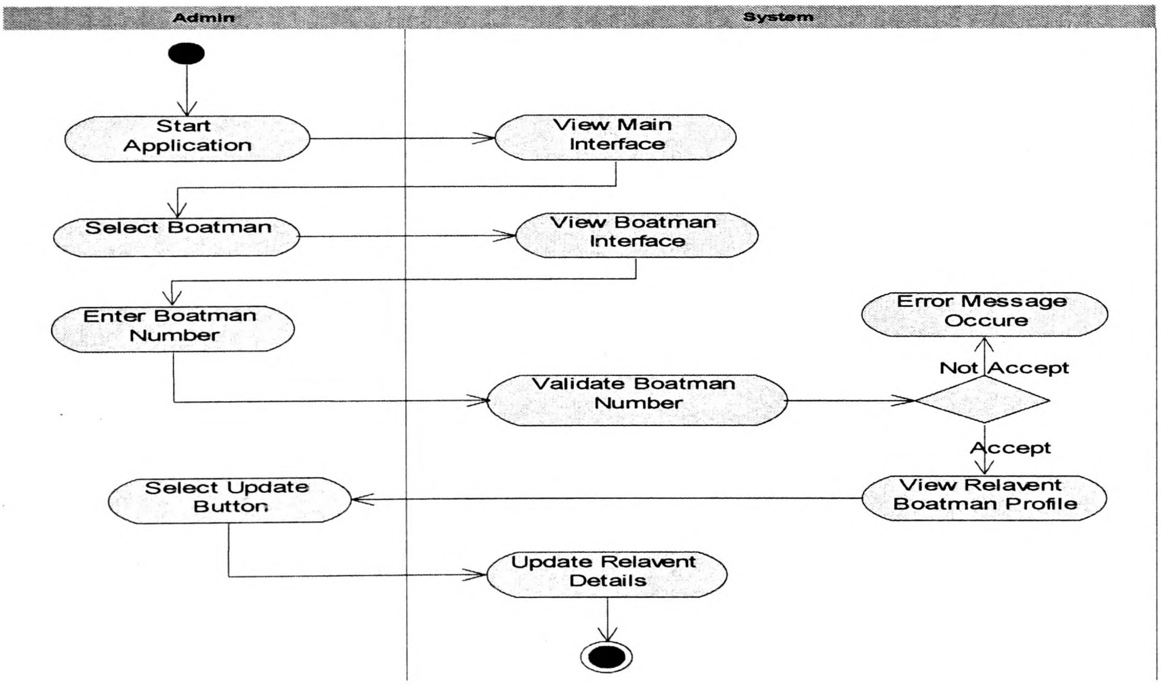
Delete User Account



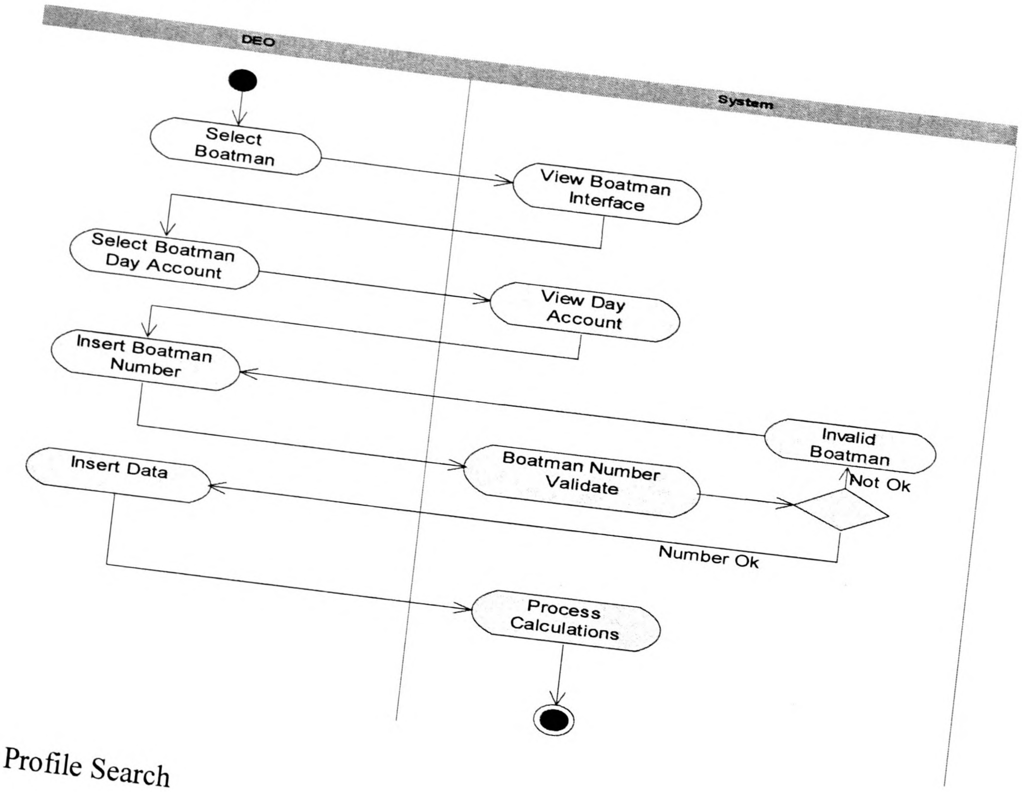
Boatman Profile Search



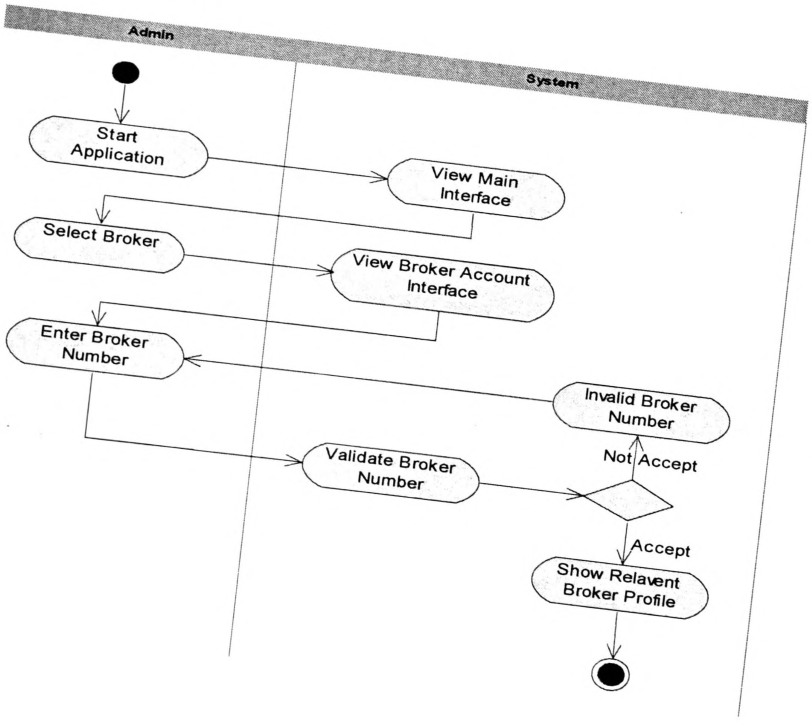
Boatman Profile Update



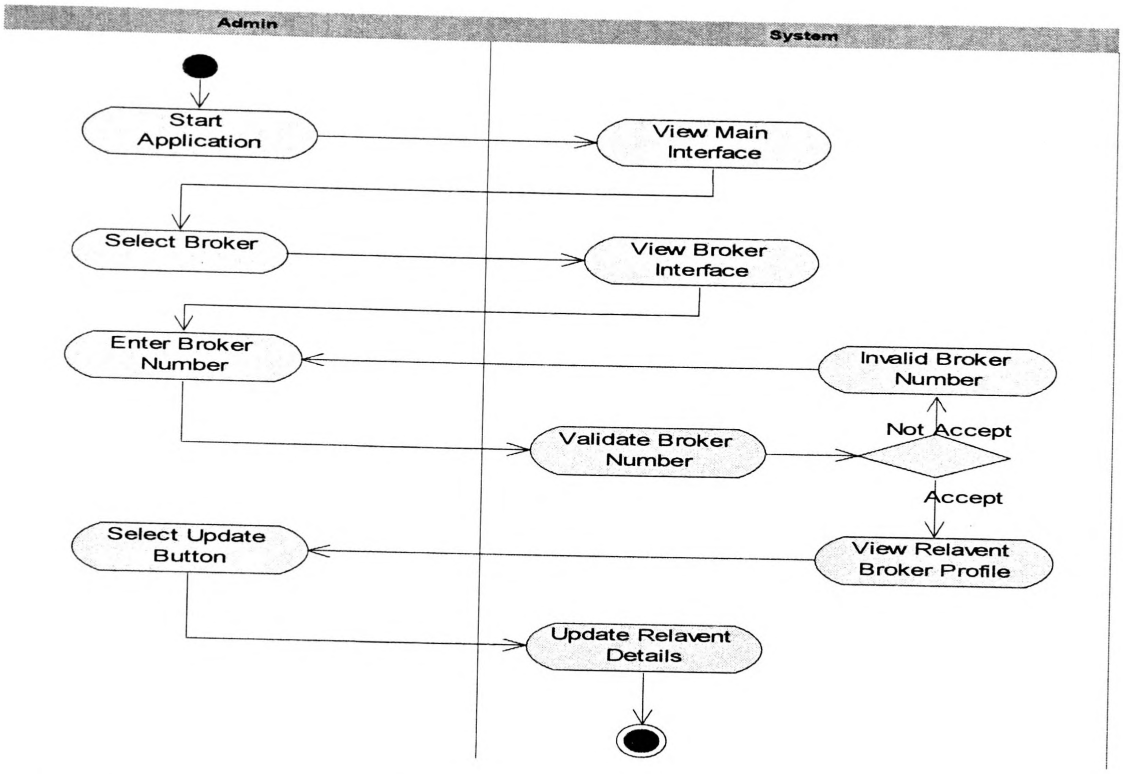
Boatman Day Account



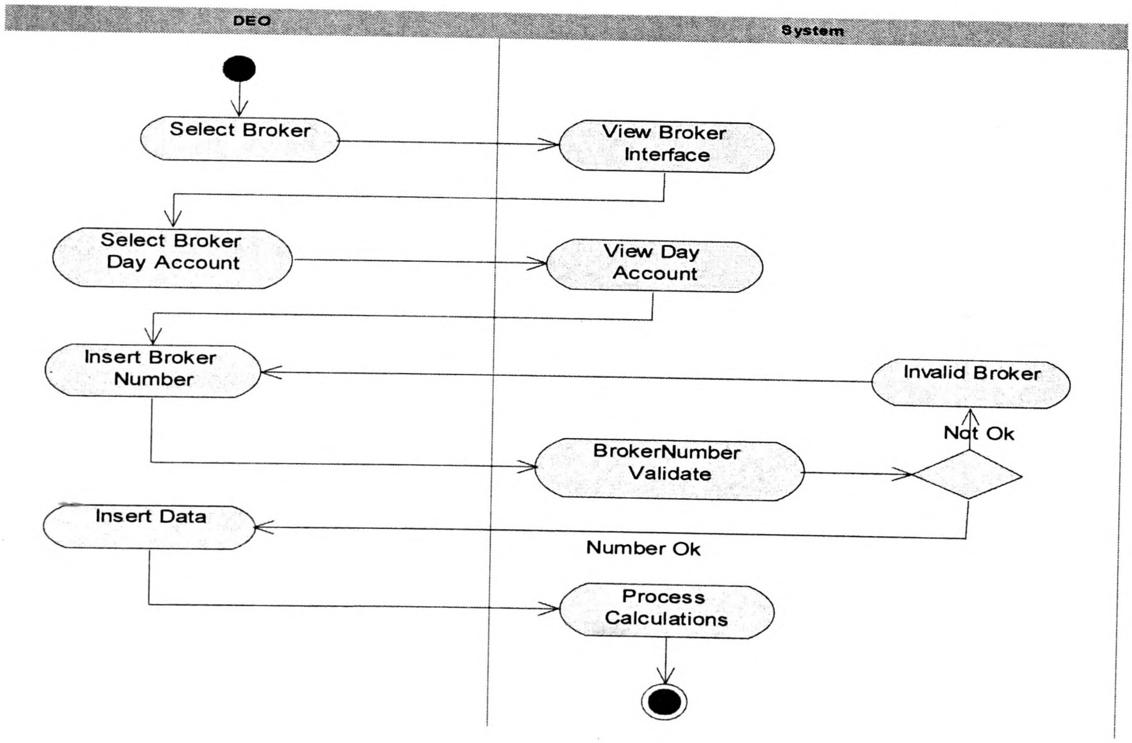
Broker Profile Search



Broker Profile Update

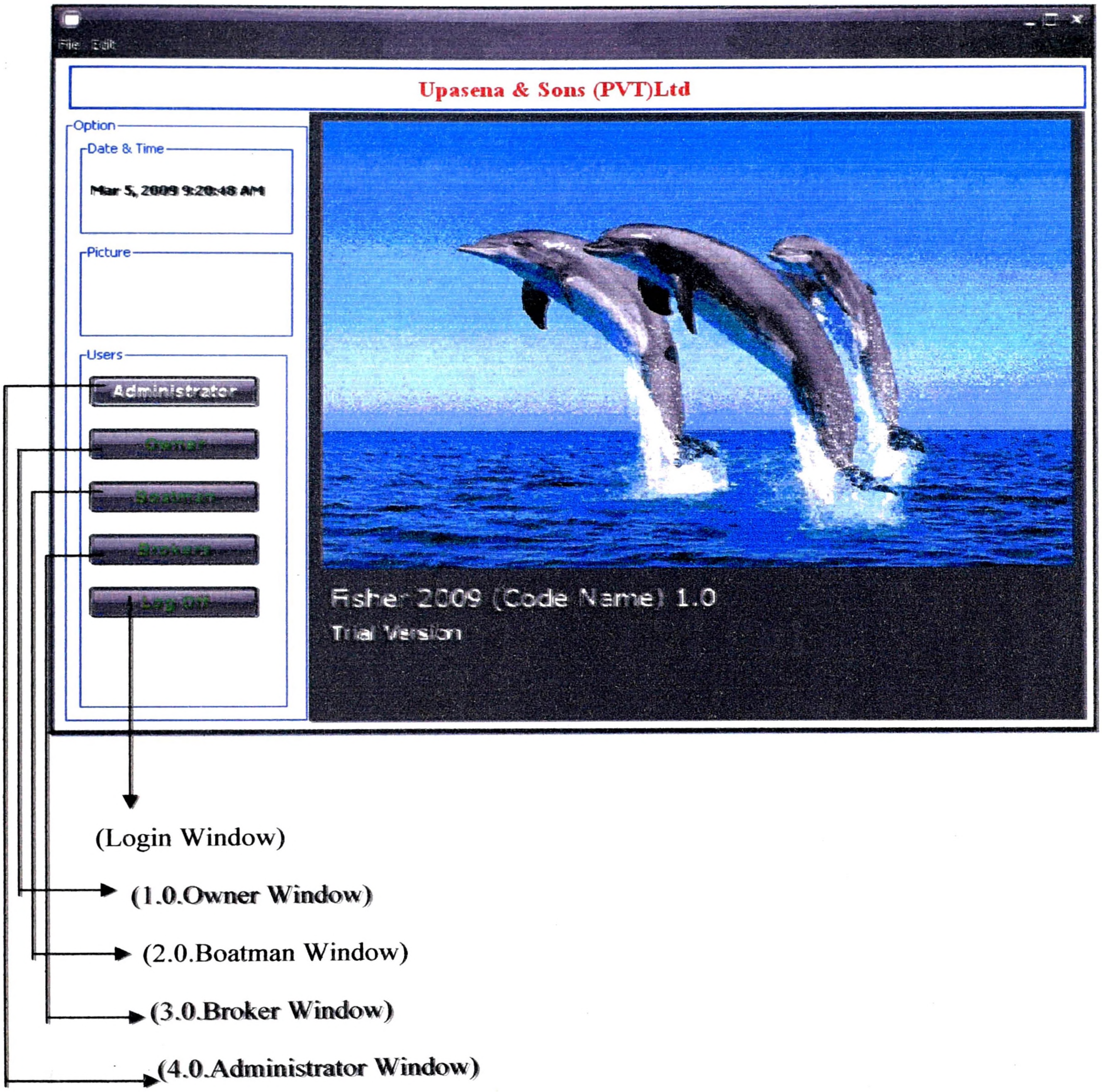


Broker Day Account

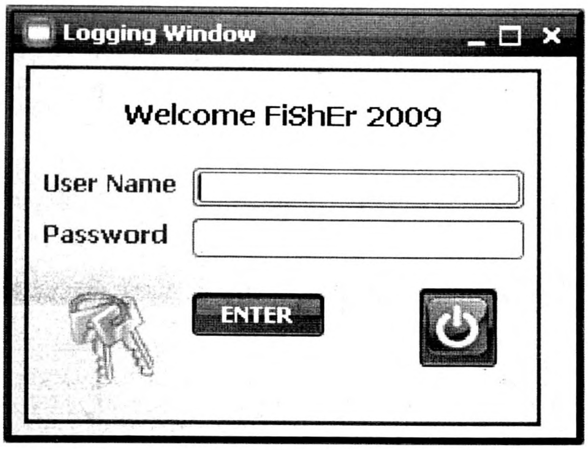


9.0. Screen Design

Main Screen

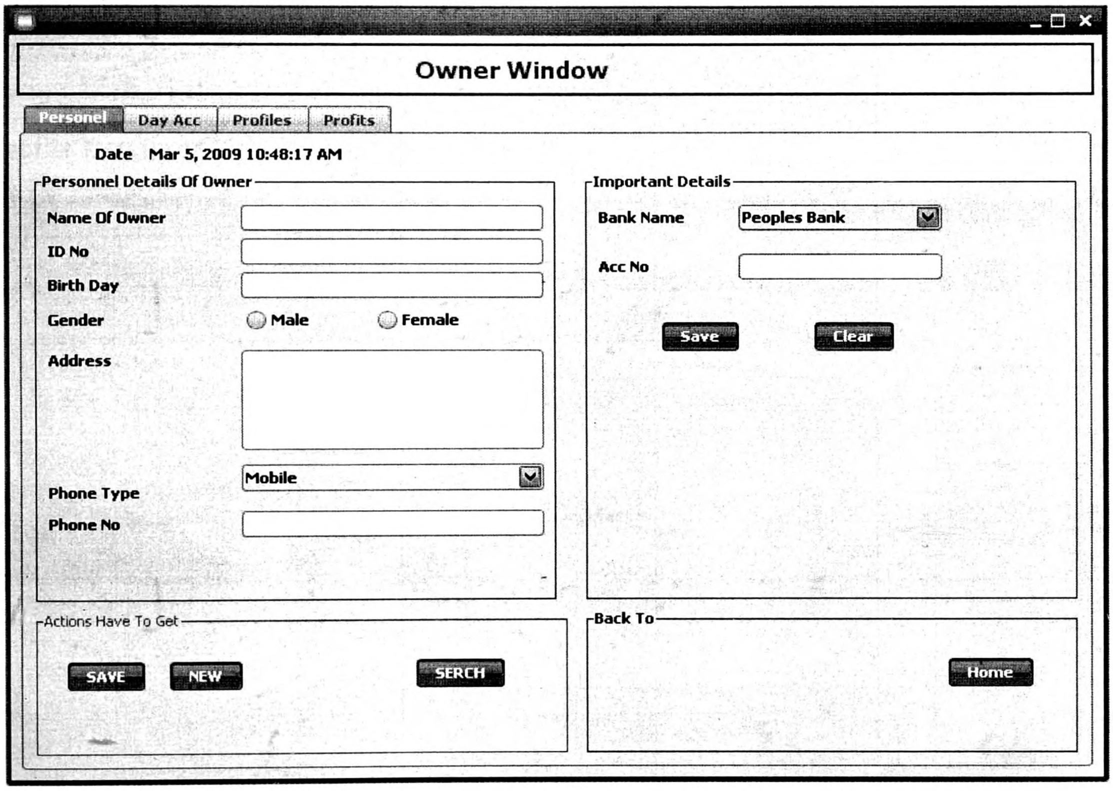


Login Window



Owner Window

Personnel Details of Owner

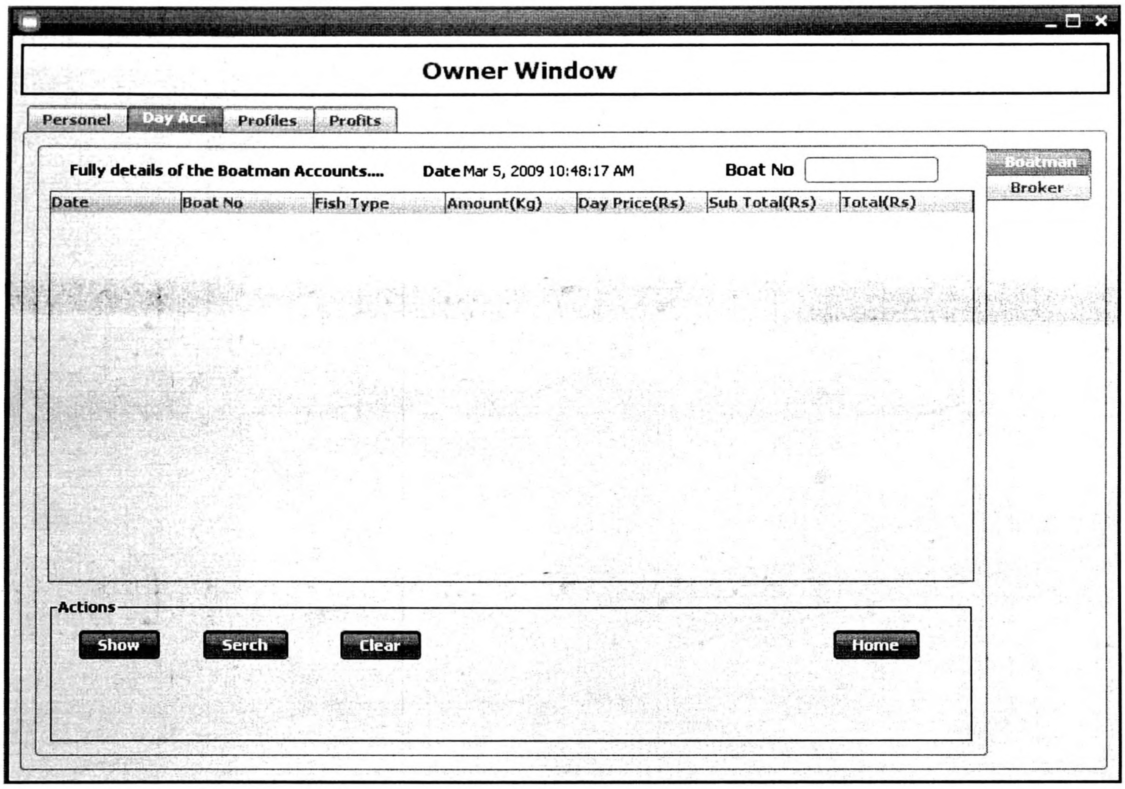


Date Mar 5, 2009 10:48:17 AM

Phone Type: Mobile

Bank Name: Peoples Bank

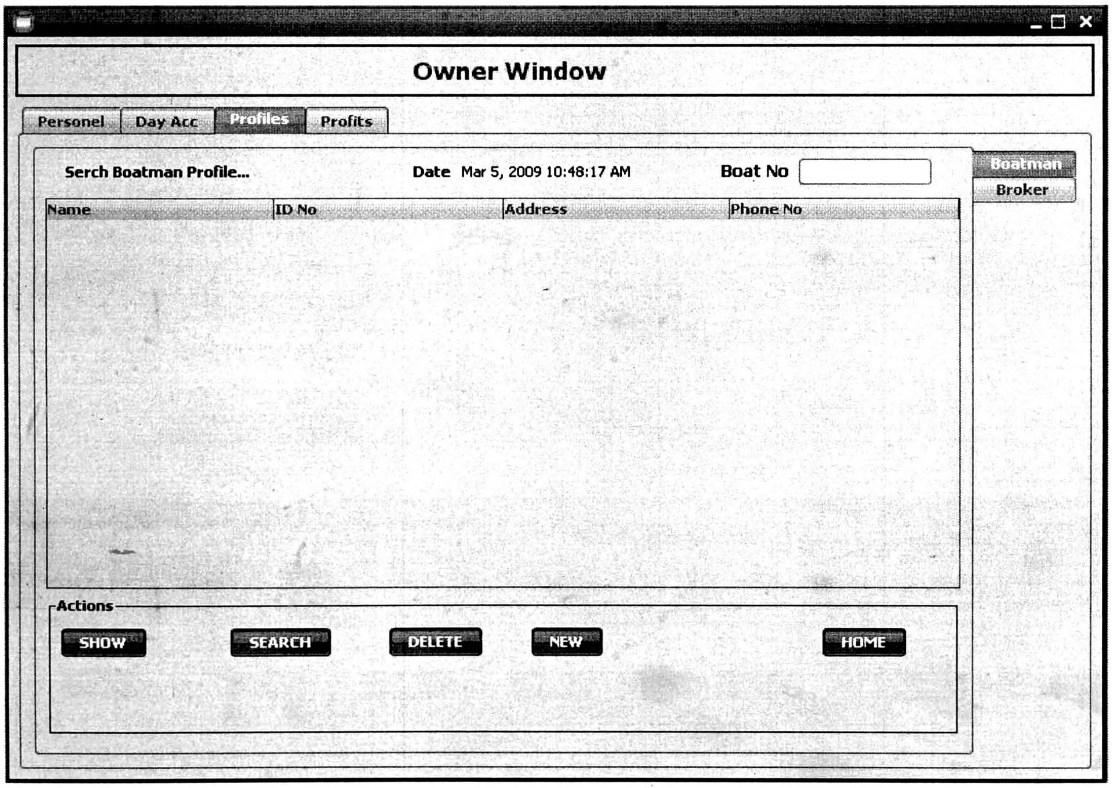
Day Account details of Owner



Fully details of the Boatman Accounts.... Date Mar 5, 2009 10:48:17 AM Boat No

Date	Boat No	Fish Type	Amount(Kg)	Day Price(Rs)	Sub Total(Rs)	Total(Rs)

Profile Details of Boatman & Broker

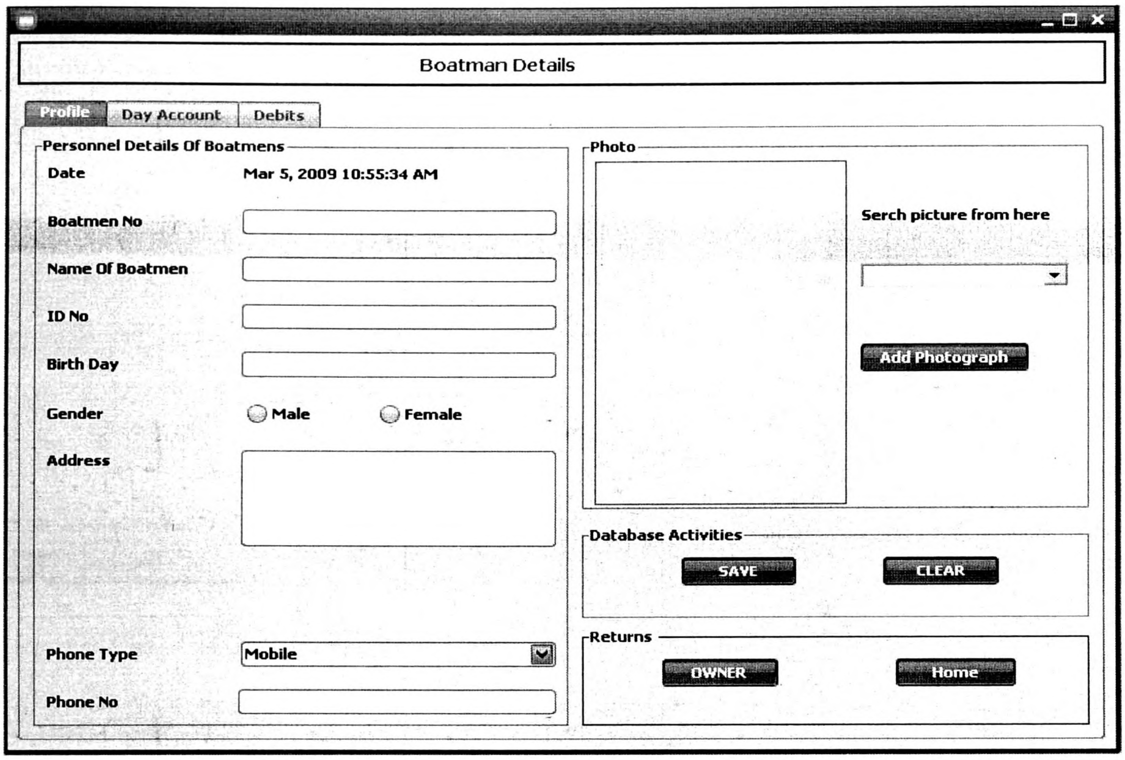


Serch Boatman Profile... Date Mar 5, 2009 10:48:17 AM Boat No

Name	ID No	Address	Phone No

Boatman Window

Boatman Personnel Details



Boatman Details

Profile | Day Account | Debits

Personnel Details Of Boatmens

Date Mar 5, 2009 10:55:34 AM

Boatmen No

Name Of Boatmen

ID No

Birth Day

Gender ○ Male ○ Female

Address

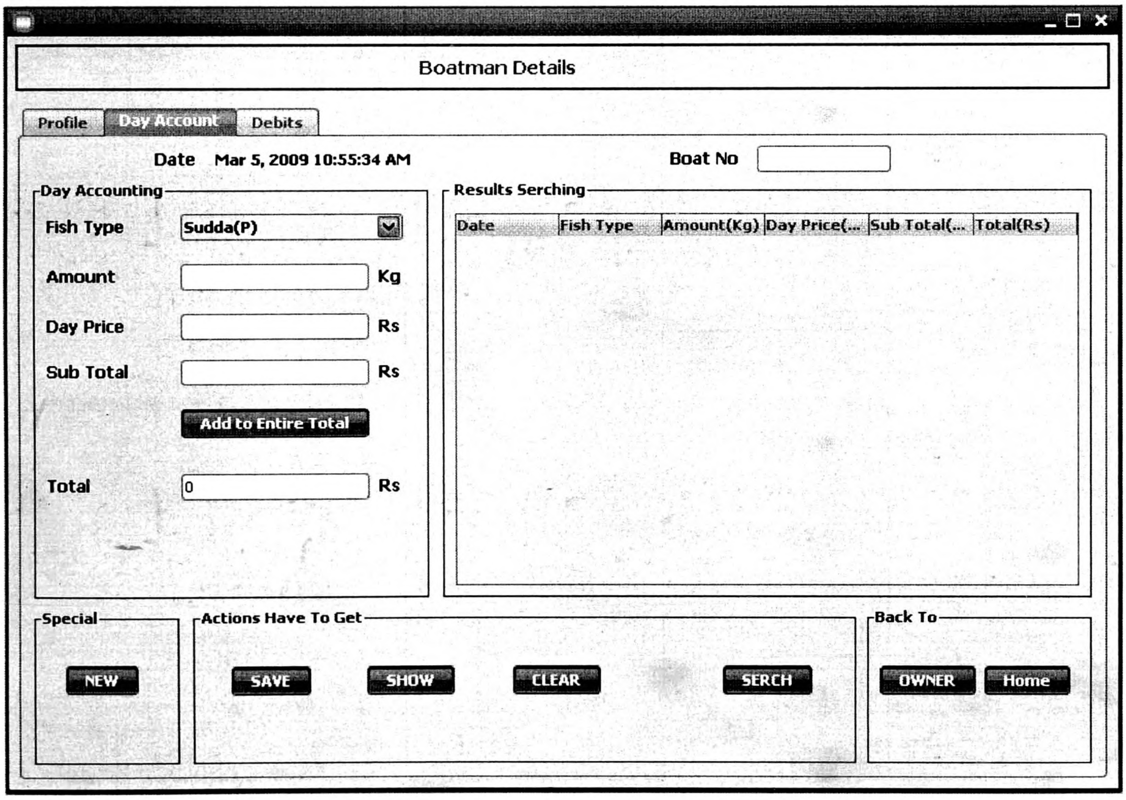
Phone Type Mobile

Phone No

Photo

Serch picture from here

Boatman Day Account Details



Boatman Details

Profile | Day Account | Debits

Date Mar 5, 2009 10:55:34 AM

Boat No

Day Accounting

Fish Type Sudda(P)

Amount Kg

Day Price Rs

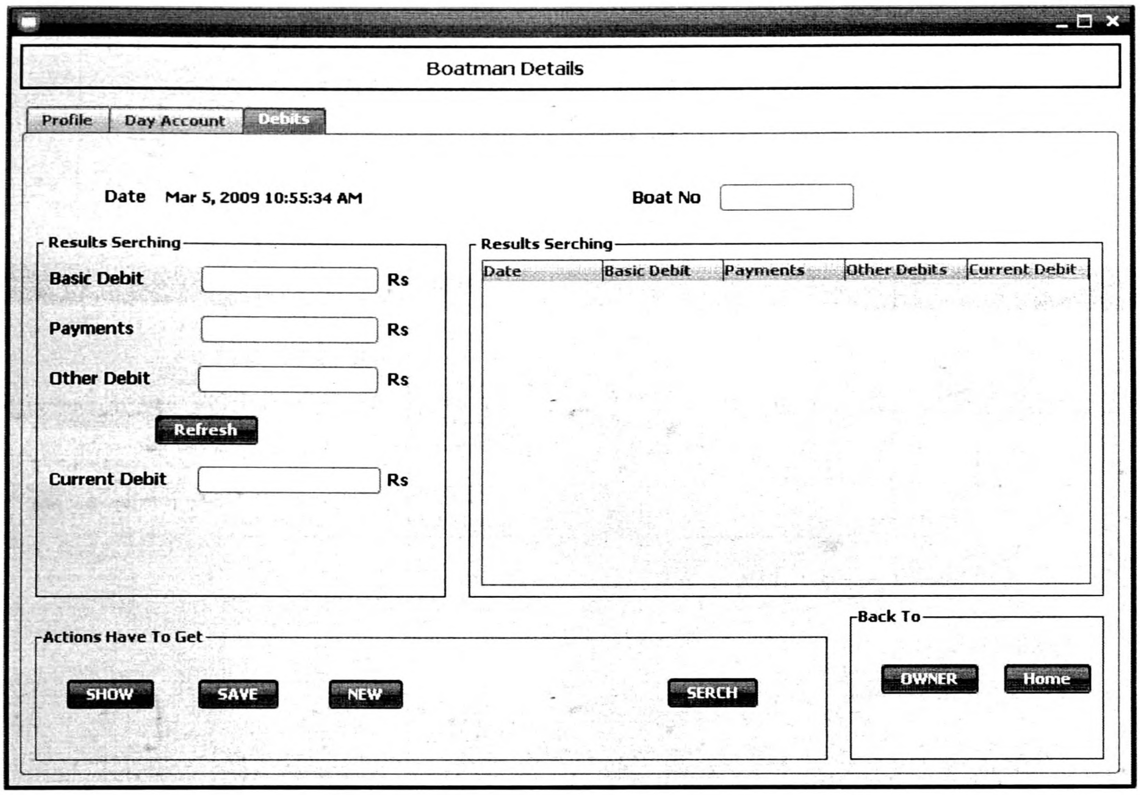
Sub Total Rs

Total 0 Rs

Results Serching

Date	Fish Type	Amount(Kg)	Day Price(...	Sub Total(...	Total(Rs)

Boatman Debit details



Boatman Details

Date Mar 5, 2009 10:55:34 AM

Boat No

Results Serching

Basic Debit Rs

Payments Rs

Other Debit Rs

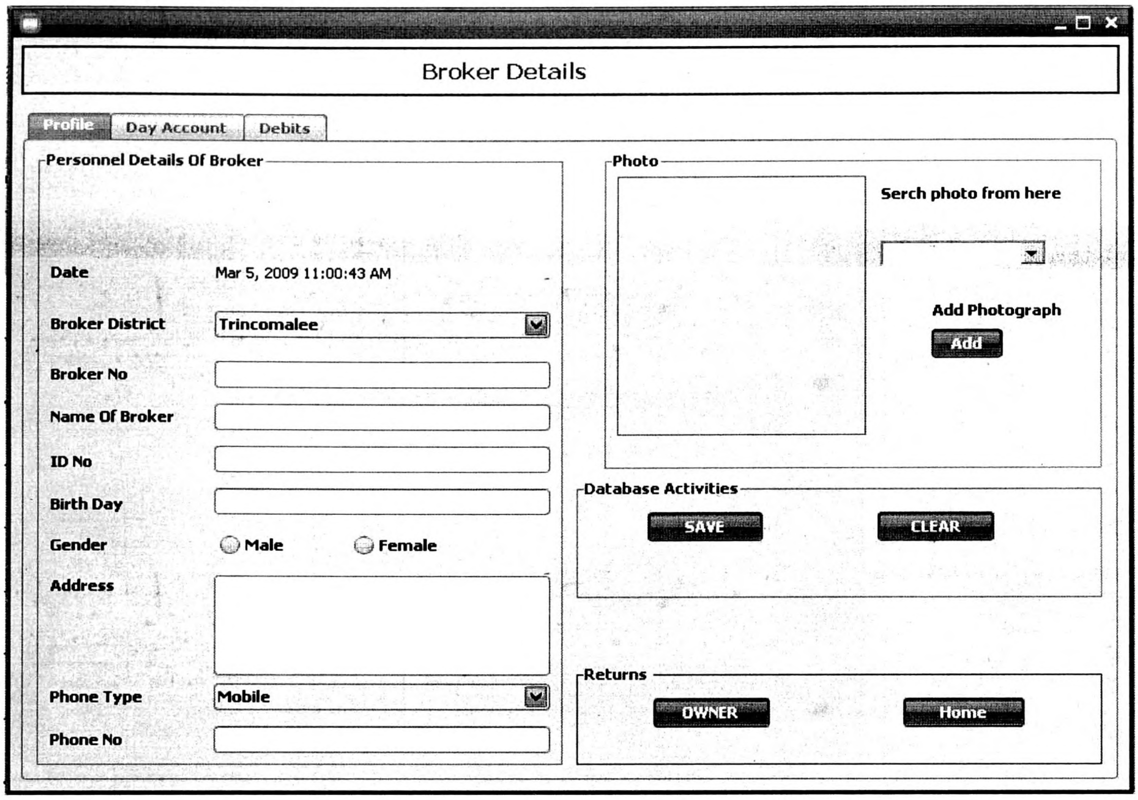
Current Debit Rs

Results Serching

Date	Basic Debit	Payments	Other Debits	Current Debit

Broker Window

Broker Personnel Details



Broker Details

Personnel Details Of Broker

Date Mar 5, 2009 11:00:43 AM

Broker District Trincomalee

Broker No

Name Of Broker

ID No

Birth Day

Gender ○ Male ○ Female

Address

Phone Type Mobile

Phone No

Photo

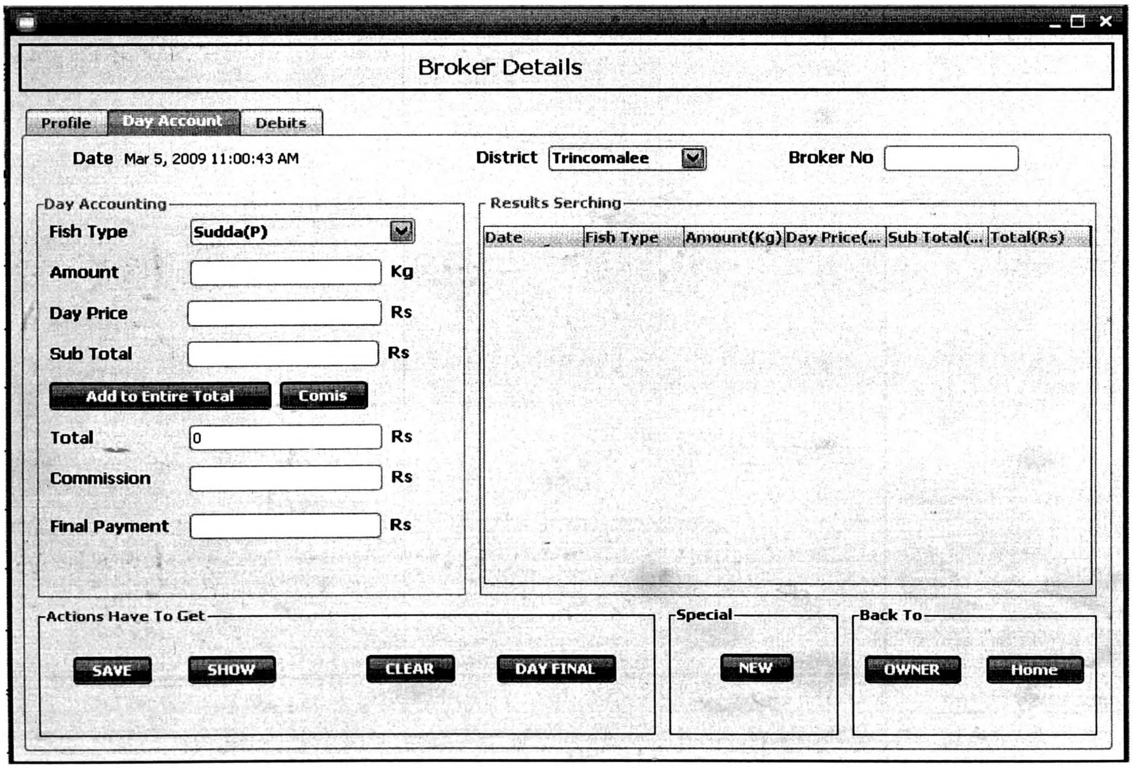
Serch photo from here

Add Photograph

Database Activities

Returns

Broker Day Account Details



Broker Details

Date Mar 5, 2009 11:00:43 AM District Trincomalee Broker No

Day Accounting

Fish Type Sudda(P)

Amount Kg

Day Price Rs

Sub Total Rs

Total 0 Rs

Commission Rs

Final Payment Rs

Results Serching

Date	Fish Type	Amount(Kg)	Day Price(...	Sub Total(...	Total(Rs)

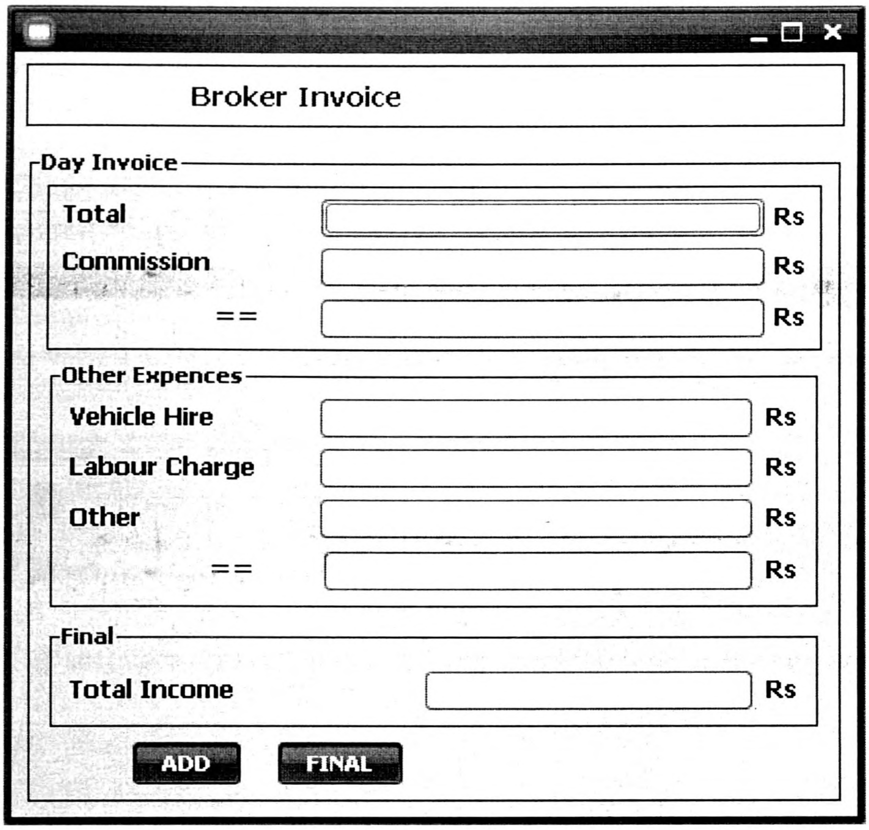
Actions Have To Get

Special

Back To

Broker Day Final

Account sheet



Broker Invoice

Day Invoice

Total Rs

Commission Rs

== Rs

Other Expences

Vehicle Hire Rs

Labour Charge Rs

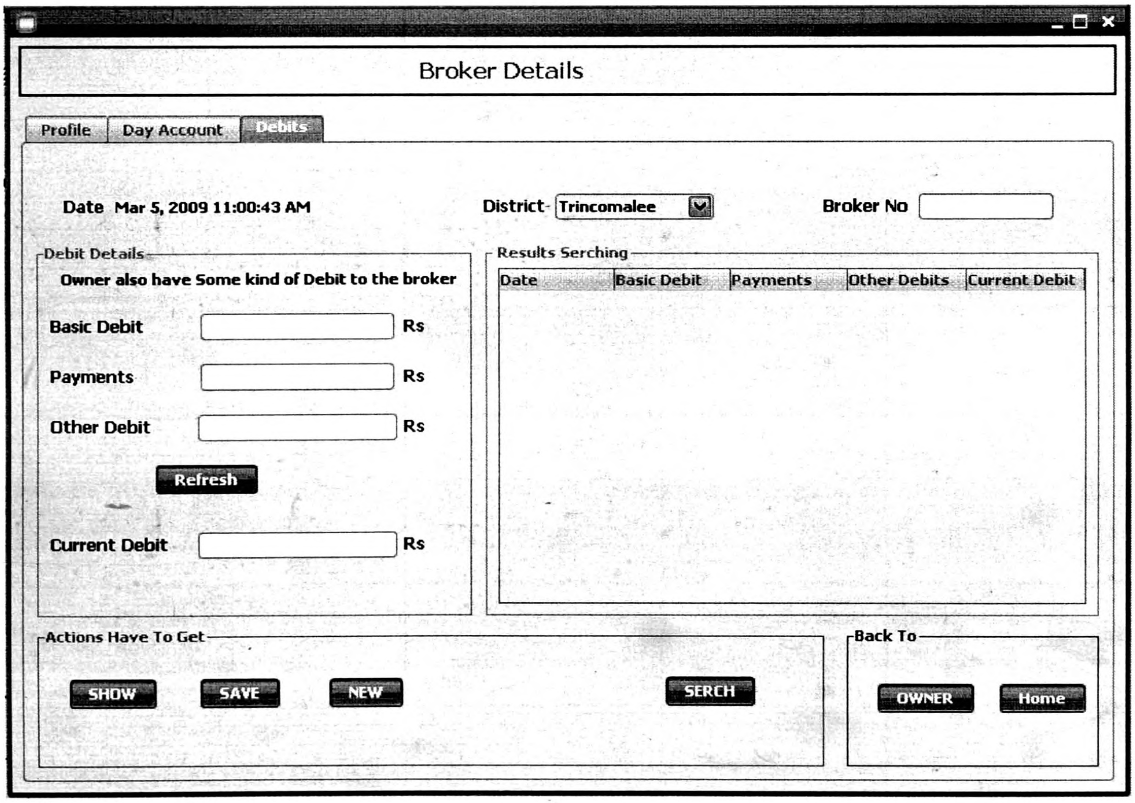
Other Rs

== Rs

Final

Total Income Rs

Broker Debit Details



Broker Details

Date Mar 5, 2009 11:00:43 AM

District Trincomalee

Broker No

Debit Details

Owner also have Some kind of Debit to the broker

Basic Debit Rs

Payments Rs

Other Debit Rs

Current Debit Rs

Results Serching

Date	Basic Debit	Payments	Other Debits	Current Debit

Actions Have To Get

Back To

10.0 Limitations

- Single PC software – Though it is a multi user multi tasking software but this can be used by single user at one time.
- The product code scenario has been predefined by the software development team and it cannot be defined by the user in the run time environment.
- The product code range has been limited according to the software team the range cannot be varied according to the user in the run time environment.
- Though it can generate standard accounting statements it is not able to generate cash flow statement.
- The profit margin cannot be generated by quarter but it can be generated by year only.
- The software doesn't make any sense about some analyzing areas.

11.0 Future enhancements

- Multi PC software – Because this is a multi user multi tasking software we are willing to enhance this to a multi PC software which can be used by many users at one time.
- We intend to generate cash flow statement.
- We are planning to generate the profit margin according to the quarter basis while performing the year basis.

12.0 Conclusion

The FISHER2009 System designed, is the end result of a lot of dedication, hard work and commitment on a part of all our group members. However this system was a dream before it could actually be implemented. Nevertheless this system would be a great value to all Financial Sector of Fishing Industry. This project has taught us a valuable lesson.

References

- Paul R. Reed, JR. 2000. Developing Applications with CORE JAVA and UML. Addison Roger Cadenhead, Inc.
- Michael McKelvy. 1997. MCSD:JAVA 6 Desktop Applications Study Guide.BPB Publications
- Summerville. 1995. The fifth edition of Software Engineering. Addison Wesley Publishers in autumn, pp.210-400
- http://www.w3schools.com/sql/default.asp
- http://www.w3schools.com/sql/sql_tryit.asp
- http://www.w3schools.com/sql/sql_functions.asp
- Coding Information, http://sourceforge.net/