Regis University

ePublications at Regis University

Regis University Student Publications (comprehensive collection)

Regis University Student Publications

Spring 2007

Small Business Ticketing System

Mohita Kadur
Regis University

Follow this and additional works at: https://epublications.regis.edu/theses

 Part of the Computer Sciences Commons

Recommended Citation

Kadur, Mohita, "Small Business Ticketing System" (2007). *Regis University Student Publications (comprehensive collection)*. 301.
https://epublications.regis.edu/theses/301

This Thesis - Open Access is brought to you for free and open access by the Regis University Student Publications at ePublications at Regis University. It has been accepted for inclusion in Regis University Student Publications (comprehensive collection) by an authorized administrator of ePublications at Regis University. For more information, please contact epublications@regis.edu.

Regis University
School for Professional Studies Graduate Programs
Final Project/Thesis

Disclaimer

Use of the materials available in the Regis University Thesis Collection ("Collection") is limited and restricted to those users who agree to comply with the following terms of use. Regis University reserves the right to deny access to the Collection to any person who violates these terms of use or who seeks to or does alter, avoid or supersede the functional conditions, restrictions and limitations of the Collection.

The site may be used only for lawful purposes. The user is solely responsible for knowing and adhering to any and all applicable laws, rules, and regulations relating or pertaining to use of the Collection.

All content in this Collection is owned by and subject to the exclusive control of Regis University and the authors of the materials. It is available only for research purposes and may not be used in violation of copyright laws or for unlawful purposes. The materials may not be downloaded in whole or in part without permission of the copyright holder or as otherwise authorized in the "fair use" standards of the U.S. copyright laws and regulations.

Running head: Small Business Ticketing System

Small Business Ticketing System

Mohita Kadur

Regis University Master of Science in Computer Information Technology

Regis University

School for Professional Studies Graduate Programs

MSCIT Program

Graduate Programs Final Project/Thesis

Disclaimer

Use of the materials available in the Regis University Thesis Collection ("Collection") is limited and restricted to those users who agree to comply with the following terms of use. Regis University reserves the right to deny access to the Collection to any person who violates these terms of use or who seeks to or does alter, avoid or supersede the functional conditions, restrictions and limitations of the Collection.

The site may be used only for lawful purposes. The user is solely responsible for knowing and adhering to any and all applicable laws, rules, and regulations relating or pertaining to use of the Collection.

All content in this Collection is owned by and subject to the exclusive control of Regis University and the authors of the materials. It is available only for research purposes and may not be used in violation of copyright laws or for unlawful purposes. The materials may not be downloaded in whole or in part without permission of the copyright holder or as otherwise authorized in the "fair use" standards of the U.S. copyright laws and regulations.

Note: The following Disclaimer is for your information. You do not need to submit it with your project. It will be scanned into all student works that are available on the WWW.

Project Abstract

Many small businesses in the cable and Internet industry provide services around cable, Internet and telephone for local residential areas. This is a win-win for both customers and service providers. Customers enjoy localized services and customer support that enables a quick turn around time on problems or outages. In order to efficiently service these customers, a ticketing system would be needed that enables customer identification, issue identification and tracking, provide updates on the status of an issue by technicians.

This project is aimed at creating such a solution for a small sized service provider business. The end result of this project would be an online system that can be used by a mini call center. The design takes into consideration the technician, administrator and call center agent view of the ticket resolution system. Each view is designed for a specific user, based on functionality described in the workflow below. This project uses Java and database technologies to implement the design.

Acknowledgements

I would like to gratefully acknowledge the enthusiastic supervision and guidance from Prof Douglas Hart during this work and project. I thank Cory Graham for the guidance and relevant discussions that helped in structuring the thesis.

I would also like to acknowledge my colleagues from the school of professional studies at Regis University, Bola Owolabi, Merry Tomling and Lencha Erena who reviewed and provided very valuable feedback on the content of this thesis. With their assistance, a third angle view was provided and helped with the betterment of the overall thesis.

I would also like to thank the School of Professional Studies and Regis University for providing useful tools and sample material that helped in the study, project and thesis.

Finally, I am forever indebted to my parents, relatives and friends for their understanding, endless support and encouragement when it was most required.

Project Paper Revision and Versioning

Version number	Comments	Author	Date
1	Initial draft	Mohita Kadur	12/17/2006
2	Submission of Draft 1	Mohita Kadur	1/11/07
3	Submission of Draft 2	Mohita Kadur	2/5/07
4	Submission of Draft 3	Mohita Kadur	2/18/07

Table of Contents

REGIS UNIVERSITY ……1

Certification of Authorship of Professional Project Work……...2

Authorization to Publish Student Work ……....4

Advisor/Professional Project Faculty Approval Form……..6

Disclaimer……..8

Project Abstract……... 9

Acknowledgements……10

Project Paper Revision and Versioning……11

Table of Contents ……....12

List of Tables……...14

List of Figures……15

1. Chapter 1 ……17

1.1 Thesis Statement……17

1.2 Problem Statement……17

1.3 Project goal……18

1.4 Need for the project……...19

1.5 Assumptions and limitations……20

1.6 Project outcomes……...21

1.7 Scope of project……22

2. Chapter 2 ……23

2.1 Literature overview and research……..23

2.2 What is known about a ticketing system……...24

2.3 Commercially available ticketing systems……29

2.4 What is unknown about a ticketing system……...31

3. Chapter 3……33

3.1 Research methods……. 33

3.2 Life-cycle models……..33

3.3 Design methodology and specific procedures…….. 36

3.3.1 Java design pattern……. 36

3.3.2 Bridge pattern…….37

3.3.3 Factory pattern……... 38

3.3.4 Model View Controller architecture……..38

3.3.5 Explanation of the design workflow……..43

3.3.6 Use case diagrams……..44

3.3.6.1 Fully dressed use cases for the call center agent as the actor……. 45

3.3.6.2 Fully dressed use cases for the administrator as the actor……..55

3.3.6.3 Fully dressed use cases for the technician as actor…….63

3.4 Database schema for the issue tracking system…… 68

3.5 Snap shots……..... 70

3.6 Review of deliverables……..... 73

3.7. Outcomes……. 73

4. Chapter 4……..**74**

4.1 How the project began……74

4.2 How the project was managed……75

4.3 Changes to the project plan…….77

4.4 Evaluation of whether project met goals……78

4.5 What went right and what went wrong……...78

4.6 Finding/Analysis of results…….79

5. Chapter 5……..**81**

5.1 What was learnt from the project experience…….81

5.2 What could have been done differently……..81

5.3 Improvements/Evolution of the project……..83

5.4 Conclusions…….84

References……...**85**

Appendixes…… **87**

A. Definition of terms ……....**87**

B. Comparison of some of the available issue tracking systems……....**92**

List of Tables

1. Customer Service Agent Workstation Requirements…….69

2. Project Task Plan……72

List of Figures

1. MVC Diagram……38

2. Design Workflow……...39

3. Use Case for Call Center Agent……42

4. Use Case for Administrator……...52

5. Use Case for Technician……...59

6. Database schema……...65

7. Login Page……67

8. Call Center Agent Login Page……..67

9. Administrator Login Page……68

10. Technician Login Page……..68

1. Chapter 1

1.1 Thesis Statement

Small scale applications service providers like Internet service or cable TV providers need a mechanism for providing a ticketing system for their customers. This system needs to be geared only to the functionality required by them and needs to be relatively in-expensive. The professional project will be geared towards the design, implementation and testing of a ticketing system for a small scale applications service provider business.

1.2 Problem Statement

In today's technically savvy world, customer service is of prime importance. Many large scale businesses have a lot of budget allocated for customer relationship management (CRM) systems. By the same token, small and medium businesses that offer valuable customer service to a select set of customers need to be able to provide the same level of customer service but at a lower budget. To cater to this need, a custom application can be designed for specific requirements as opposed to buying commercially available software that tend to be more pricy and have features that the small business does not use. Based on the research done on the Internet and questions asked of professionals working in the creation of call center software, it can be concluded that several applications exist that are either free of cost and provide different functionality for such requirements. These applications tend to be mainly built and meant for the larger business. Even though companies try to market software for the mid-market business, they tend to have many features built in that are not really needed by all small businesses.

A custom solution tends to be inexpensive and efficient way to proceed because only the required features will be incorporated into the software design. Also from a project development, training and maintenance perspective, it would work out cheaper only to focus on the functionality that the specific business utilizes as opposed to any other additional functionality that comes built in a commercial software package.

1.3 Project Goal

A specific example that fits this scenario for a project would be a small service provider business that installs hardware like high speed Internet, cable TV and Voice over IP telephone service to a specific set of local zip codes. Customers are benefited by a local service provider that can service their accounts. The advantages include personalized service, faster turn around for problems incurred and potentially lower cost to have this service. To be able to correctly track customers, the services installed, the problems reported and resolution of the problem, a ticketing system can be used. This kind of CRM system benefits both the customer and the service provider. The CRM tracking system will provide features such as: Customer identification in order to be able to initiate the tracking process and uniquely classify issues based on specific customers, contact details of a customer in order to maintain communication medium and in some cases provides an alternative form of identifying the customer, services installed so that the call center agent can review and identify which services have an issue that needs to be resolved, current status of the customer in order to identify active status of their account, nature, severity of the problem in order to assign the ticket status or escalate accordingly.

1.4 Need for the project

This project labeled as “small business customer relationship management ticketing system”, can be generically categorized as an issue tracking system. An issue tracking system (ITS) is a software application that allows an enterprise to record and follow the progress of every problem or "issue" that a computer system user identifies until the problem is resolved. With an ITS, an "issue", which can be anything from a simple customer question to a detailed technical report of an error or bug, can be tracked by priority status, owner, or some other customized criteria.

An ITS generally provides the user with a way to report an issue, track progression towards its resolution, and know who is responsible for resolving the issue. It also allows the manager of the system to customize the tracking procedure so that unnecessary documentation on the part of the problem solvers does not become a waste of time. Many kinds of enterprises use ITS applications, including software developers, manufacturers, IT help desks, and other service providers.

Issue tracking provides the following benefits such as the ability to chart the progress of individuals and the overall resolution of the issue, it also provides the ability to track and maintain end to end history of the issue in order to providing the call center agent a comprehensive view of the actions that have taken place till date. Another advantage of having issue history is to be able to identify the type of issues that the customer has faced in the past and the pattern if any exists. Another benefit of issue tracking is the ability to generate a report for the service provider to get a full view of tickets and any filters that are being searched for. These reports also enable the senior management to get a view of

the work performed and in case of any escalations from a customer, it would be easy to run through these reports to get details of the case and appropriately handle the situation.

1.5 Assumptions and limitations

This project is geared mainly towards a student's master's project. Based on this, the commercial usability of the end software will depend on the following assumptions and limitations:

The requirements being developed against are restricted to a limited functionality based on a relatively small sized business. The average size of such a business would be about 20 field technicians and about 5 to 6 customer service representatives. The project itself is geared towards a single office location as opposed to multiple sites. The mode of operation will be server side and database components will be installed on a server. All client end i.e. customer service agents, technicians, administrators etc will have a browser based client to access the application and perform daily operations.

The operation of the software does not consider any failover or redundancy deployment. Hence in case of failures, downtime will be observed. This project's primary objective is geared towards learning and implementation of technologies such as JAVA, java servlets, web based GUI and database technologies etc using Java Design Patterns and Model View Controller architecture. The approach was to keep the requirements to a minimal functional ticketing system in order to provide sufficient development, testing and documentation time. This system can be improved in various ways and made more feature rich as indicated in the enhancements section.

At any given time there will be only one open ticket per customer per problem area and access to the database is synchronized. Customers reporting a problem are already registered with the retail shop.

1.6 Project objectives

The objectives on the professional project would include the following: Create, submit and attain approval on the project proposal from faculty, decide on requirements for the professional project, input to this phase will come from experience, subject matter experts on the topic. A decision will be made on the platform and software to use to develop the software. Input to this decision will be biased by the technology of comfort and software that is generally available and is efficient for the development of this software. Based on the design suggested, approval will be attained from the project advisory board. Once the design has been approved, development of the modules based on the requirements and design will take place. As the modules are developed, unit testing will be performed for each module to determine functioning as per design and specifications. As more and more modules are developed, inter-functionality of these modules will be tested using system and integration testing techniques. There are no formal tools that will be used to perform this testing. For the most part, the testing performed will be manual. Once the project development is completed, it will be analyzed and compared with the design specifications that were set at the beginning of the project. Final approval on the project implementation will be attained and any feedback incorporated into the implementation as necessary.

1.7 Scope of Project

The small business ticketing system will be built on requirements got from industry best standards as well as communicating with an industry expert that deals with developing call center software. The requirements are limited in order to provide a basic functioning ticketing system and to be able to complete development of the solution in the time frame available. All the components of the project would be server side components and once deployed would require very minimal modifications. From a user perspective, with minimal training of the system, users that are Internet savvy would be able to use the system. The goal of the project is to provide users like technicians, call center agents and administrators a web interface to use the system. The system in also designed to be flexible and technologies chosen e.g. Java design pattern etc can enable addition of extra functionality as required.

2. Chapter 2

2.1 Literature Overview and Research

A trouble ticket (sometimes called a *trouble report*) is a mechanism used in an organization to track the detection, reporting, and resolution of some type of problem. Trouble ticketing systems originated in manufacturing as a paper-based reporting system, now most are Web-based and associated with customer relationship management (CRM) environments, such as call centers or e-business Web sites, or with high-level technology environments such as network operations centers. A number of companies make software for trouble ticketing, such as NesterSoft's *Request Commander*. Several other types of software, such as Bluebird include a trouble ticket component.

The Internet Engineering Task Force's Network Working Group specified requirements for a trouble ticketing system in RFC(Request For Comments) 1297 (NOC Internal Integrated Trouble Ticket System Functional Specification Wish list). In the RFC (Request For Comments)document, the author compares the trouble ticket to a patient's hospital chart, because both define a problem and help to coordinate the work of several different people who will work on the problem at different times to solve it.

As a ticket moves though the system, it is usually classified as a certain type of issue, which in turn determines the skill set and expertise level of the agent(s) the ticket is assigned to. Until the issue is resolved, the "open ticket" for the problem remains in the work queue, with issues of highest priority taking precedence in terms of work flow. [1]

2.2 What is known about a ticketing system

A typical workflow involved in a ticket resolution process could include the following steps: [2]

Step 1: Recording the Problem

The problem may be found by an internal or external person. There are two ways that a problem can come from the external person: by phone or by email. For the internal person, there must be a screen that can be called up without delay that presents a form for entering the details of the problem. Submitting this form will cause the creation of the workflow process, and at the same time generate a unique ID for the trouble ticket.
A phone call from an external person will be handled by an internal person, who uses essentially the same form mentioned above, but needs to be able to search for registered customers a couple of different ways. Again, submission of the form will create the process and assign an id. The ID of the trouble ticket must be produced by the system and be immediately available (within 10 seconds) in order to let the external person know the trouble ticket ID. The ID is used as a way to call up the trouble ticket when that person calls in again to check on progress.

Finally, email may be sent to a particular address. This email is automatically picked up, and a trouble ticket started, which includes the body of the email as part of the data. No attempt is made to automatically analyze the body of the message, but the sender's email address is retrieved from the header. The first step of the process is for some internal person to read the message, and to fill in some of the other fields on the form with real values, and then submit it.

The system should be able to look up the registered user from the email address. When the trouble ticket is submitted, a email confirmation is sent to the external person, and the process goes to step 2.

Step 2: Reproduce the Problem

This step is designed to check the trouble ticket report, and to see if it describes accurately a reproducible problem. This activity is simply to follow the instruction on the report, and to see if the described behavior occurs. If the trouble ticket comes from an internal person, then this activity must be assigned to someone else so that the recording and reproducing of the problem are not done by the same person. If it comes from an external person, this activity may be done by the same person who enters the report. If the behavior can not be reproduced, then this process goes to step3, otherwise it goes to step 4.

The problem may be identified at this stage. If there is a known solution to the problem it should be entered or referred to at this stage, and then communicated back to the originator by going to Step 6. If the problem is recognized as a duplicate of another problem, it should be able to be recorded as such, and go to step 5.

Step 3: Correcting the Report

This step is reached only if the problem can not be reproduced. This step is assigned to the originator if internal. If external, this must be assigned to a person who can contact the originator and get more clarification on the problem. There are two results of this step, either back to Step 2, or to give up on the process and go to step 6.

Step 4: Identifying the Problem and Resolution

This is where the specialist is called in. The problem details should narrow down the area of the problem. If the expert determines that the area is wrong, it should be able to be reassigned, and the person assigned to the activity should change.

The problem stays in this state until a resolution is determined. Either the problem is identified and it will be fixed, or it be fixed later due to schedule constraints, or it is determined to be a misunderstanding and is actually the correct behavior. In all cases the resolution must be communicated to the originator, either via email, or else through a phone call. It goes to step 5.

For this organization, accomplishing this activity require invoking a sub process. The development team has its own workflow process that handles this in a manner that fits the way they work. The exact route of this sub process is not the subject of this scenario, only that it is started, it is given data, and at some time later it reports that it is complete and returns a set of data.

The sub process for the development team was implemented before the trouble ticket scenario, so it already has a set of field with meaning appropriate to that task. This means that the trouble ticket process must translate the fields into the field used by the sub process. The details of this are defined below.

Step 5: Verifying the Resolution

When the problem is resolved, then it waits for the resolution to become available. When it is available, the resolution is verified. If the resolution was "fixed," and the problem is not actually solved, then the process can be sent back to step 4. Otherwise the process goes to step 6.

Step 6: Communicate Results

The results of the process are communicated back to the originator here. This step contains a rule that the result must be communicated within 3 days of being known. If not, an email message is sent to the support manager.

Step 7: Audit and Record

This step may happen before or after Step 6, but must happen before the end of the process. It involves someone looking at the process and determines whether the question/answer should be included in a monthly newsletter to the user community. This step is started in parallel with Step 6 since it does not depend upon it, and might happen before it.

The above process explains a typical call center ticket resolution process for a trouble caused and handled by a customer relationship management system. This project focuses on the ticket management aspect of the process. This ticket management system facilitates all the above mentioned process and provides up to date data irrespective of whichever customer service representative handles the issue. It also maintains a log of the issues occurred for documentation purposes.

The most common issue tracking system's design is relatively simple. A database is the main storage repository for all data. These data are managed by the business logic layer of the application. This layer gives the underlying raw data more structure and meaning, preparing it for human consumption. The now human readable data are then presented to the support technician by another software application or web page. The end-user of the issue tracking system can create entirely new issues, read existing issues, add details to existing issues, or resolve an issue. Anytime a user of the system makes a change, the

issue tracking system will record the action and who made it, so as to maintain a history of the actions taken. Each user of the system may have issues assigned to them, that is, that user is responsible for the proper resolution of that issue. This is generally presented to the user in a list format. The user may have the option of re-assigning an issue to another user, if needed.

Issues can have several aspects to them. Each issue in the system may have an urgency value assigned to it, based on the overall importance of that issue. Critical issues are the most severe that should be resolved in the most expedient way possible, taking precedence over all other issues. Low or zero urgency issues are minor, and should be resolved as time permits. Other details of issues include the customer experiencing the issue (whether external or internal), date of submission, detailed descriptions of the problem being experienced, attempted solutions or work-around, and other relevant information. As previously noted, each issue maintains a history of each change.

Examples of data items that can be used to track customer items in a ticketing system are: Originator UID - a unique id if one exists, Originator Name, Originator Address, Originator phone, Originator email address, Submitter - the person who took the call. If internal, same as originator, Synopsis - a one line description, Description - a full in depth description, Source Email - holds the email that started the process, if there is one, Severity, Priority, Product, Area, Date Received (Submitted), Date Resolved, Date Verified, Date Closed, Attached data files (URLs to files checked into a server), Expert - the person who is the expert for the current product area, Resolution, Resolution Description, Status (presumably part of the workflow...), Date of last originator contact and Duplicate Ticket number.

2.3 Commercially available ticketing systems

Here are some research examples of small business based CRM applications and how they relate to the small business ticketing system built.

Entellium eSalesForce small CRM Sales Force Automation delivers features like opportunity management, shared calendaring, forecasting and collaborative tools. The CRM service is delivered over the internet for businesses.

This system is mainly focused on a sales organization. Some of the features that are advertised around this system include easy customization, no need of a database, secured transactions and easy upgrade.

This is a serviced based product and offering is a sales based application. It is meant for a small sales based business that does not want to deal with the hassle of hosting a server based application, maintenance, up-keep etc. These kind of applications have a per user, per period based fee for the service. All the technical support, up-keep, upgrade, customizations etc are performed by the service provider.

In comparison to the small business ticketing system developed, it is targeted towards a support based organization. It is targeted towards an organization that can invest upfront in the hardware to deploy the application and not incur any additional per user or per period fees. In terms of features, it is targeted towards ticketing systems for problem resolution, update and tracking of a support ticket as opposed to a sales organization.

Microsoft Dynamics CRM is a customer relationship management (CRM) solution that provides the tools and capabilities needed to create and easily maintain a clear picture of customers, from first contact through to purchase and post-sales. With modules for sales, marketing, and customer service, Microsoft Dynamics CRM delivers a fast, flexible, and affordable solution that drives consistent, measurable improvements in every business process, enabling closer relationships with customers and helping to achieve new levels of profitability.

This software is a very powerful CRM tool that has been built with the capability of addressing requirements from various businesses like sales, marketing, customer service etc. It has a flexible workspace that has customizable GUIs; it has in-built reporting systems and has built in integrations to Microsoft products like Outlook etc. Some of the other benefits that are listed around this software offering are ease of modification, fast data searches, ease of use, ease of customization and flexible reporting.

Deployment costs of this application include license fees as well as professional services to deploy the solution.

Although it says it is meant for a small and medium business, it seems a little expensive after all the costs are taken into consideration. It is a really feature rich system but in many cases some small businesses that operate only in a support framework would not want to pay for all the other features that are sales or marketing oriented that they would not use anyways. Also with the deployment and installation costs in addition to the per user costs come annual maintenance costs. All said and done the price point of such applications are not very attractive to business that customer service and tracking systems are at a low scale.

Salesforce.com is another popular CRM application that many call centers use commercially. It comes in various flavors like sales, marketing, service and support. For the intents of this project, the focus of analysis will be only on the service and support part of the software.

Salesforce.com provides features like contact management using proactive contact, interactive voice response or self-service applications, email, chat, consolidated reporting etc.

Salesforce.com is also a hosted model with pay as you go pricing. For a medium business that uses all the features listed above, this model works out really well as up front investment in hardware and ongoing maintenance and staff to keep the system updated is not needed.

As listed in the above statements, a very small service provider that works on low budget may not need all the features listed and hence may not afford these systems.

Considering the above examples, the small business system that has been developed may not have all the features that many commercial systems have but considering a very specific need that a service provider has and the price that they may be willing to pay, with this in mind the project was developed. It can be extended based on requirements as they come up.

2.4 What is unknown about a ticketing system

Initially at the beginning of the project, it was unclear if a thin web based client was to be used or a thick based client. Based on further research, it was decided that the solution was going to be deployed on a single server and users access it via a browser and hence a

thin client was chosen. As this application is not meant in its current state to be deployed in a live production environment, the costs of software like the web server, database servers etc are not taken into consideration.

The scalability of the solution is also unknown. The testing for this solution won't involve stress or load tests and the number of users that this system is capable of handling is unknown.

Based on some research done on commercially available issue tracking systems and common attributes, the table shown in Appendix B.

3. Chapter 3

3.1 Research Methods

During the course of this project, the Internet was used to understand how ticketing systems that are commercially available were designed. A subject matter expert was consulted on a weekly basis to understand how the design compares to a full fledged commercially available ticketing system. As a master's project, the primary goal of the requirements being designed is closer to a proof of concept. Although the requirements have a lot of thought process, the functionality provided caters to only the bare necessities of a service provider business. Most of the commercial software have a lot of extra functionality that are packaged in order to meet generic features of many users. An advantage of this type of approach with commercial software is that many of the features that are not used right away are available for future use. The advantage of the approach taken in this project is to develop requirements based on the specific need and this tends to keep the cost of software low and easy to afford by the smaller business.

3.2 Life-Cycle Models

The systems development life cycle (SDLC) is a conceptual model used in project management that describes the stages involved in an information system development project, from an initial feasibility study through maintenance of the completed application. Various SDLC methodologies have been developed to guide the processes involved, some of the more common ones are the waterfall model , rapid application development , the spiral model, build and fix; and synchronize-and-stabilize.

In general, an SDLC methodology follows the following steps:

1. The existing system is evaluated and deficiencies are identified.

2. The new system requirements are defined

3. The proposed new system is designed.

4. The new system is developed and tested to reveal any errors and the necessary changes are made.

5. Deploy the new system in the production environment.

6. The system is deployed in the production environment, then re-evaluate and maintain the new system

Each of the methods mentioned was compared in order to select the appropriate method to use for this project. The Waterfall Method is deemed to be one of the original methods developed. It is also said to be the least desirable, because there is no allowance for revisiting previous steps.

The Waterfall Method follows the general steps, and as such, does not begin a later step, until the previous step is completed. (Definitions) This method only allows for user input during the evaluation and testing phases. It is less useful in dynamic environments, but is still a common method.

The Rapid Application Development (RAD) method does not necessarily evaluate an existing system. (Definitions) Instead, there is generally a session with users to gather requirements, and then a prototype is built. The prototype may use and reuse the existing

system. This methodology is seen as a better fit for object oriented programming. Once the users have approved of the prototype, the actual system is designed based on this.

The Spiral Model is much like the Waterfall method, in that it follows the same steps. The difference is that the Spiral Model takes into account that there may be a need to revisit previous steps. (Definitions) In order to compensate for this, it is actually a series of shorter Waterfall Methods. The steps are completed once to design a prototype, and then steps are followed a second or multiple times until a satisfactory prototype is designed, and the project moves into the final system design. This method is preferred for larger, more expansive projects, as it also allows for the project to be terminated if it is deemed to be cost-ineffective or unnecessary for any other reason.

The Build and Fix method is just as it sounds. It is a very basic method that does not incorporate planning. (Definitions) Rather, a system is programmed, delivered to the user, and modified as the user requests. This is a very risky method, since it does not allot for planning that may reveal the project to be undesirable.

The Synchronize and Stabilize method incorporates portions of the Spiral Model and Waterfall Method, but allows for various departments to work on the project simultaneously. This method allows for planning, and also allows for flexibility in the system development. This method has been used by many of the big software developers, such as Microsoft and Netscape.

Based on the review of these methodologies, the size and nature of this project indicated the Synchronize Method to be the most appropriate. The goal of this project is to develop an issue tracking system for a small retail business center to provide a better tracking for all customer problems and customer service experience. Although the

synchronize method states that different departments can work in parallel on the project, this was the approach used on the project. Different modules were developed in parallel. One of the reasons that the parallel approach was to accommodate project timelines and to be able to deliver the required functionality on time. Additionally, some of the base functionality of the modules was identical up to a point and this made it easy to begin the modules in parallel and then change the development based on the specific nature of the module. Based on this reasoning, the Synchronize and Stabilize Method was chosen as the SLDC method.

3.3 Design Methodology and Specific Procedures

In the next few paragraphs, description and explanation of technologies, procedures and implementation strategies used in this project are provided. These technologies were chosen for the design of the project as they increase scalability, flexibility and portability which are best practices of software development. The model view controller approach promotes scalability and helps to keep the user interface, the controller and the business modules as separate entities. The design using the factory and bridge pattern promotes code re-use. Flexibility and portability is introduced in the design by the ability to change very minimal code in order to support other platforms hence requiring very minimal design or code modification.

3.3.1 Java Design Pattern

If a problem occurs over and over again, a solution to that problem has been used effectively. That solution is described as a pattern. The design patterns are language-

independent strategies for solving common object-oriented design problems. When you make a design, you should know the names of some common solutions. Learning design patterns is good for people to communicate with each other effectively. In fact, you may have been familiar with some design patterns, you may not use well-known names to describe them. SUN suggests GOF (Gang of Four--four pioneer guys who wrote a book named "Design Patterns"- Elements of Reusable Object-Oriented Software), so we use that book as guide to describe solutions. In the project to be implemented the bridge pattern and the factory pattern will be used as a design methodology.

3.3.2 Bridge Pattern

The bridge pattern is a design pattern used in software engineering which is meant to “decouple an abstraction from its implementation so that the two can vary independently”. The bridge uses encapsulation, aggregation and can use inheritance to separate responsibilities into different classes.

When a class varies often, the features of object oriented programming become very useful because changes in program’s code can be made easily with minimal prior knowledge about the program. The bridge pattern is useful when not only the class itself varies but often also what the class does. The class itself can be thought of as the abstraction and what the class can do as the implementation.

In the project implementation there are 2 interfaces that extent the marker interface and 2 classes that implement these interfaces. One of the interfaces contains all methods required for adding a new ticket, updating an existing ticket and resolution of the problem. The other interface contains all methods needed for adding an employee,

deleting an employee and updating an employee. The reasons for using the bridge pattern are to separate abstraction and implementation permanently, share an implementation among multiple objects, want to improve extensibility and hide implementation details from clients

3.3.3 Factory Pattern

A Factory pattern is one that returns an instance of one of several possible classes depending on the data provided to it. Usually all of the classes it returns have a common parent class and common methods, but each of them performs a task differently and is optimized for different kinds of data.

In the project the factory is used to generate the required object at runtime based on the input given and returns a reference to an interface. Based on the type of interface returned, the necessary method in invoked, which is implemented in the class that implements the interface.

3.3.4 Model View Controller Architecture

In implementing the issue tracking system the MVC architecture was used which is a way of breaking an application, or even just a piece of an application's interface, into three parts: the model, the view, and the controller. MVC was originally developed to map the traditional input, processing, output roles into the GUI realm:

Input-->Processing-->Output

Controller --> Model --> View

Model

A model is an object representing data or even activity, e.g. a database table or even some plant-floor production-machine process. The model manages the behavior and data of the application domain, responds to requests for information about its state and responds to instructions to change state. The model represents enterprise data and the business rules that govern access to and updates of this data. Often the model serves as a software approximation to a real-world process,so simple real-world modeling techniques apply when defining the model.

The model is the piece that represents the state and low-level behavior of the component. It manages the state and conducts all transformations on that state. The model has no specific knowledge of either its controllers or its views. The system itself maintains links between model and views and notifies the views when the model changes state. The view is the piece that manages the visual display of the state represented by the model. A model can have more than one view.

For the application being developed the model was responsible for getting data, in the form of a bean processing it and storing it in the database. It was also responsible for getting data from the database, stuffing it into a bean and sending it to the controller.

View

A view is some form of visualization of the state of the model. The view manages the graphical and/or textual output to the portion of the bitmapped display that is allocated to its application. The view renders the contents of a model. It accesses enterprise data through the model and specifies how that data should be presented. For the project being

developed the view consisted of html and jsp that were rendered to the user either to display a result or to fill information that needed to be processed.

Controller

A controller offers facilities to change the state of the model. The controller interprets the mouse and keyboard inputs from the user, commanding the model and/or the view to change as appropriate. A controller is the means by which the user interacts with the application. A controller accepts input from the user and instructs the model and view to perform actions based on that input. In effect, the controller is responsible for mapping end-user action to application response. The controller translates interactions with the view into actions to be performed by the model. In a stand-alone GUI client, user interactions could be button clicks or menu selections, whereas in a Web application they appear as HTTP GET and POST requests. The actions performed by the model include activating business processes or changing the state of the model. Based on the user interactions and the outcome of the model actions, the controller responds by selecting an appropriate view. The controller is the piece that manages user interaction with the model. It provides the mechanism by which changes are made to the state of the model.

In the project being developed the all HTML and JSP's are sent to the servlet, which acted as the controller to package all the data from the forms that were submitted and sent it to the model for processing. The controller also received the data from the model after processing and passed it to the view to be rendered to the user.

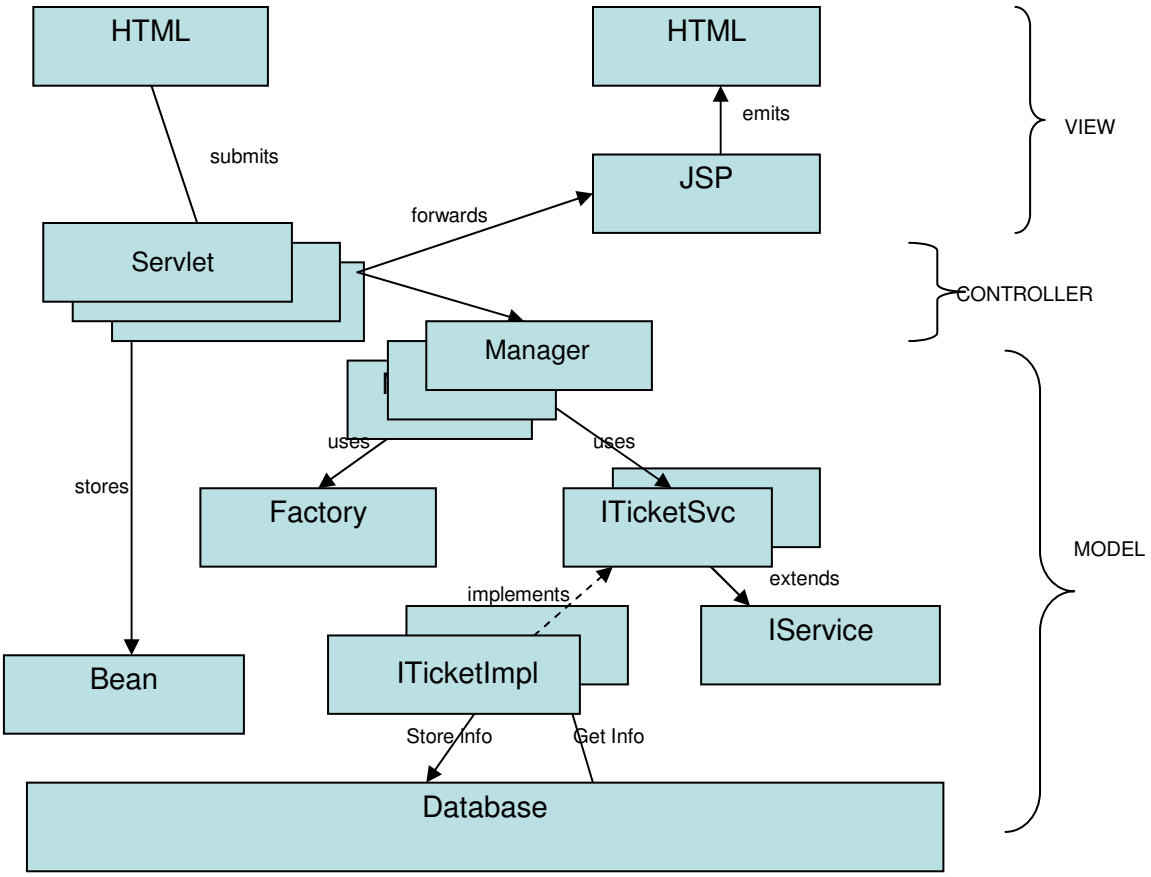


Figure 1 MVC Diagram

Design Workflow

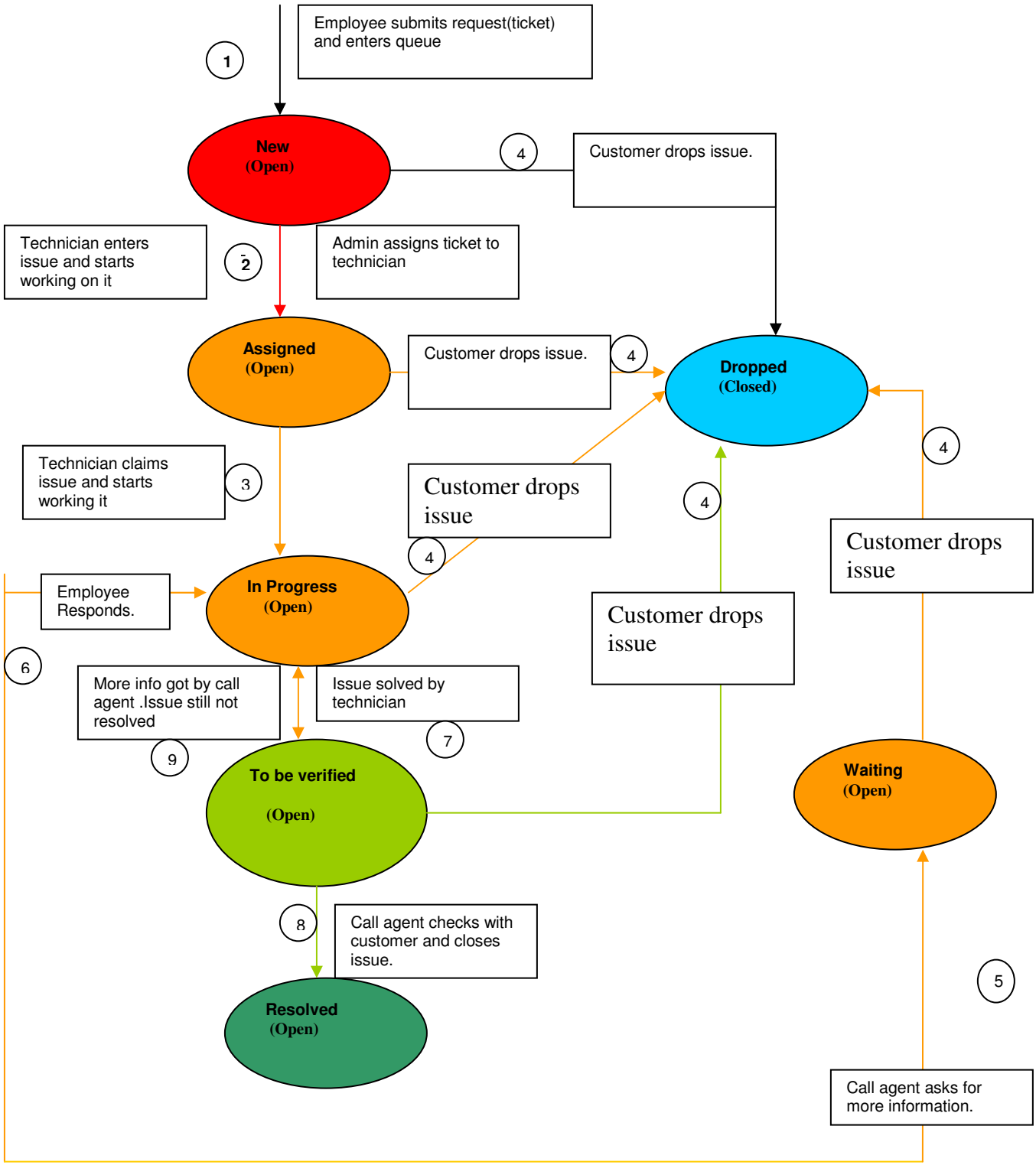


Figure 2 Design Workflow

3.3.5 Explanation of the design workflow

The explanation of all the steps that occur during the use of the issue tracking system is given below**:**

1. Employee logs in and fills out the new ticket form and submits the form. The state of the new ticket is open and progress is assigned as new.

2. The administrator logs in and opens all tickets that are new and open, and assigns the ticket to a technician based on the information in the new ticket form, now the ticket is in the open state and is assigned.

3. The technician logs in and opens all tickets assigned to him which are new and open and starts working on it. The ticket now is in the open state and in progress.

4. If a customer calls in and says problem has been fixed, the ticket is moved into the closed and dropped state.

5. If the information provided by the call center agent is not enough to solve the problem, the technician puts the ticket in the open and waiting state, and waits for more information to be collected from the customer.

6. The call center agent looks at the waiting ticket, calls the customer to get the needed information and puts the ticket into the open and progress state.

7. If the technician is able to resolve the issue, he puts the ticket into the open and to be verified state.

8. The call center agent looks at the ticket to be verified, calls the customer and verifies if the problem has been fixed and puts the ticket into the closed and resolved state.

9. The call center agent looks at the ticket to be verified, calls the customer and asks the customer if the problem got fixed ,if the problem still persists the call agent gets more information about the problem and puts the ticket back into the open and progress state.

3.3.6 Use Case Diagrams

The Use case diagram is used to identify the primary elements and processes that form the system. The primary elements are termed as "actors" and the processes are called "use cases." The Use case diagram shows which actors interact with each use case.

The above statement pretty much sums up what a use case diagram is primarily made up of—actors and use cases. A use case diagram captures the functional aspects of a system. More specifically, it captures the business processes carried out in the system.

As you discuss the functionality and processes of the system, you discover significant characteristics of the system that you model in the use case diagram. Due to the simplicity of use case diagrams, and more importantly, because they are shorn of all technical jargon, use case diagrams are a great storyboard tool for user meetings. Use case diagrams have another important use. Use case diagrams define the requirements of the system being modeled and hence are used to write test scripts for the modeled system.

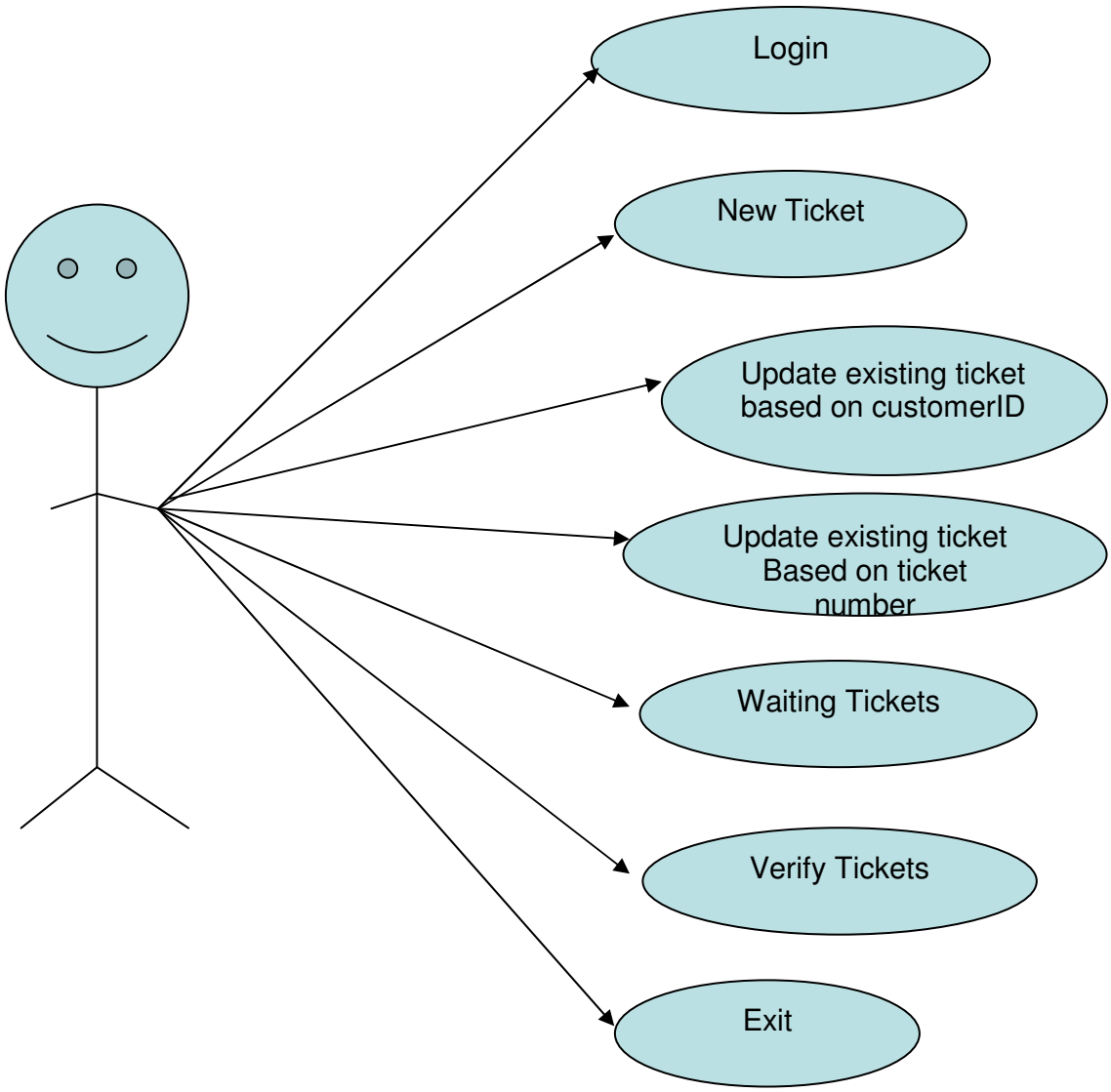


Figure 3 Use case for Call center agent

3.3.6.1 Fully dressed use cases for the call center agent as the actor

Use case: Call center agent login

Primary Actor: Call center agent

Stakeholders and interests:

Company: wants secure, reliable issue management system

Preconditions: The issue tracking system is prompting the user to enter username and password.

Success guarantee (post condition): The call center agent has access to enter new ticket or update existing tickets.

Main success scenario:

1. User enters username and password.

2. Presses the submit button.

3. If username and password is valid, the call center agent login page is displayed.

Extensions (alternate flow)

1. If username or password is invalid, the error page is displayed and user is prompted to enter the right username and password.

Use case: Call center agent, enter new ticket

Primary actor: Call Center Agent

Stakeholders and interests:

Company: wants secure, reliable issue management system

Preconditions: The issue tracking system is prompting the user to enter username and password.

Success guarantee (post condition): The call center agent fills out the new ticket information, which is stored in the database.

Main success scenario:

1. Call center agent enters username and password.

2. Presses the submit button.

3. If username and password is valid, the call center agent login page is displayed, which has 5 button options, one of which is new ticket.

4. User chooses new ticket button, which then opens up a new ticket entry form.

5. User fills out all required fields and hits the submit button.

6. Success message is displayed and user has option to go back to the call center agent login page by selecting the Go Home button.

Extensions (alternate flow)

1. If all required fields in the new ticket form are not filled, then an error message is displayed, and the user is prompted to fill all required fields.

2. If the application was unable to store the new ticket information in the database due to some system error, an error message is displayed and the user has the option of going back to the call agent home page and can try to enter the information again.

Use case: Call center agent, update existing ticket given ticket Number

Primary actor: Call center agent

Stakeholders and interests:

Company: wants secure, reliable issue management system

Preconditions: The issue tracking system is prompting the user to enter username and password.

Success guarantee (post condition): The call center agent fills out the update ticket information, which is stored in the database.

Main success scenario:

1. Call center agent enters username and password.

2. Presses the submit button.

3. If username and password is valid, the call center agent login page is displayed, which has 5 button options, one of which is existing ticket.

5. User chooses existing ticket number hyperlink, which then opens a page and prompts the call center agent to enter the ticket number and hit the submit button

6. Then a page is displayed with the existing ticket information for the given ticket number where the call agent can review the existing information.

7. The user then hits the Update button, which will take him to the update ticket form page.

8. User fills out all required fields appropriately and hits the submit button.

9. Success message is displayed and the updated ticket is stored into the database and the user has option to go back to the call center agent login page by selecting the Go Home button.

Extensions (alternate flow)

1. If all required fields in the new ticket form are not filled, then an error message is displayed ,and the user is prompted to fill all required fields.

2. If the application was unable to update the new ticket information in the database due to some system error, an error message is displayed and the user has the option of going back to the call agent home page and can try to enter the information again.

3. If the state and progress fields are not filled appropriately, then an error message is displayed, and the user is prompted to fill them appropriately.

4. If an invalid ticket number is entered, then an error message is displayed and the user has an option to go back to the home page by hitting the go home button.

Use case: Call center agent, update existing ticket given customer ID

Primary actor: Call center agent

Stakeholders and interests:

Company: wants secure, reliable issue management system

Preconditions: The issue tracking system is prompting the user to enter username and password.

Success guarantee (post condition): The call center agent fills out the update ticket information for the appropriate ticket chosen based on a display of available tickets for the given customer ID, which is stored in the database.

Main success scenario:

1. Call center agent enters username and password.

2. Presses the submit button.

3. If username and password is valid, the call center agent login page is displayed, which has 5 button options, one of which is existing ticket.

4.User chooses existing ticket customer ID hyperlink, which then opens a page and prompts the call center agent to enter the customer ID and hit the submit button

5. Then a table is displayed with a summary of all existing tickets which are open for the given customer ID.

6. User then can choose the appropriate ticket that needs to be update by selecting a radio button and hit submit

7. Then a page is displayed with the existing ticket information for the selected ticket where the call agent can review the existing information

8. The user then hits the Update button, which will take him to the update ticket form page.

9. User fills out all required fields appropriately and hits the submit button.

10. Success message is displayed and the updated ticket is stored into the database and the user has option to go back to the call center agent login page by selecting the Go Home button.

Extensions (alternate flow)

1. If all required fields in the new ticket form are not filled, then an error message is displayed, and the user is prompted to fill all required fields.

2. If the application was unable to update the new ticket information in the database due to some system error, an error message is displayed and the user has the option of going back to the call agent home page and can try to enter the information again.

3. If the state and progress fields are not filled appropriately, then an error message is displayed, and the user is prompted to fill them appropriately.

4. If an invalid customer ID is entered, then an error message is displayed and the user has an option to go back to the home page by hitting the go home button.

Use case: Call center agent, update tickets that need to be verified

Primary actor: Call Center Agent

Stakeholders and interests:

Company: wants secure, reliable issue management system

Preconditions: The issue tracking system is prompting the user to enter username and password.

Success guarantee(post condition):The call center agent gets a list of the tickets that need to be verified by him, he call the customer and based on his conversation fills out the update ticket form , which is stored in the database.

Main success scenario:

1. Call center agent enters username and password.

2. Presses the submit button.

3. If username and password is valid, the call center agent login page is displayed, which has 5 button options, one of which is verify tickets.

4. Then a table is displayed with a summary of all tickets which are open and need to be verified, so that the ticket can be closed.

5. User then can chooses one of the ticket by selecting a radio button and hits submit

6. Then a page is displayed with the existing ticket information for the selected ticket where the call agent can review the existing information

7. User will then have to call the customer and ask if the problem was fixed or not.

8. The user then hits the Update button, which will take him to the update ticket form page.

9. User fills out all required fields appropriately based on his conversation with the customer and hits the submit button.

10. Success message is displayed and the updated ticket is stored into the database and the user has option to go back to the call center agent login page by selecting the Go Home button.

Extensions (alternate flow)

1. If all required fields in the new ticket form are not filled, then an error message is displayed, and the user is prompted to fill all required fields.

2. If the application was unable to update the new ticket information in the database due to some system error, an error message is displayed and the user has the option of going back to the call agent home page and tries to enter information again.

3. If the state and progress fields are not filled appropriately, then an error message is displayed, and the user is prompted to fill them appropriately.

4. If there are no tickets that need to be verified at that time, then a message is displayed and the user has an option to go back to the home page by hitting the go home button.

Use case: Call center agent, update tickets that are in waiting state

Primary actor: Call center agent

Stakeholders and interests:

Company: wants secure, reliable issue management system

Preconditions: The issue tracking system is prompting the user to enter username and password.

Success guarantee (post condition): The call center agent gets a list of the tickets that are in waiting state; he calls the customer and based on his conversation fills out the update ticket form, which is stored in the database.

Main success scenario:

1. Call center agent enters username and password.

2. Presses the submit button.

3. If username and password is valid, the call center agent login page is displayed, which has 5 button options, one of which is waiting tickets.

4. Then a table is displayed with a summary of all tickets which are open and are in waiting state, so that the ticket can be processed by the technician.

5. User then can chooses one of the ticket by selecting a radio button and hits submit

6. Then a page is displayed with the existing ticket information for the selected ticket where the call agent can review the existing information

7. User will then have to call the customer and ask for the required information so that the technician can proceed to solve the problem.

8. The user then hits the Update button, which will take him to the update ticket form page.

9. User fills out all required fields appropriately based on his conversation with the customer and hits the submit button.

10. Success message is displayed and the updated ticket is stored into the database and the user has option to go back to the call center agent login page by selecting the Go Home button.

Extensions (alternate flow)

1. If all required fields in the new ticket form are not filled, then an error message is displayed, and the user is prompted to fill all required fields.

2. If the application was unable to update the new ticket information in the database due to some system error, an error message is displayed and the user has the option of going back to the call agent home page and tries to enter information again.

3. If the state and progress fields are not filled appropriately, then an error message is displayed, and the user is prompted to fill them appropriately.

4. If there are no tickets in the waiting state, then a message is displayed and the user has an option to go back to the home page by hitting the go home button.

Use case: Call center agent exit

Primary actor: Call center agent

Stakeholders and interests:

Company: wants secure, reliable issue management system

Preconditions: The issue tracking system is displaying the exit button on the login page.

Success guarantee (post condition): The call center agent has successfully logged out of the system.

Main success scenario:

1. User hits the exit button.
2. User has exited the application. Logged out message is displayed.

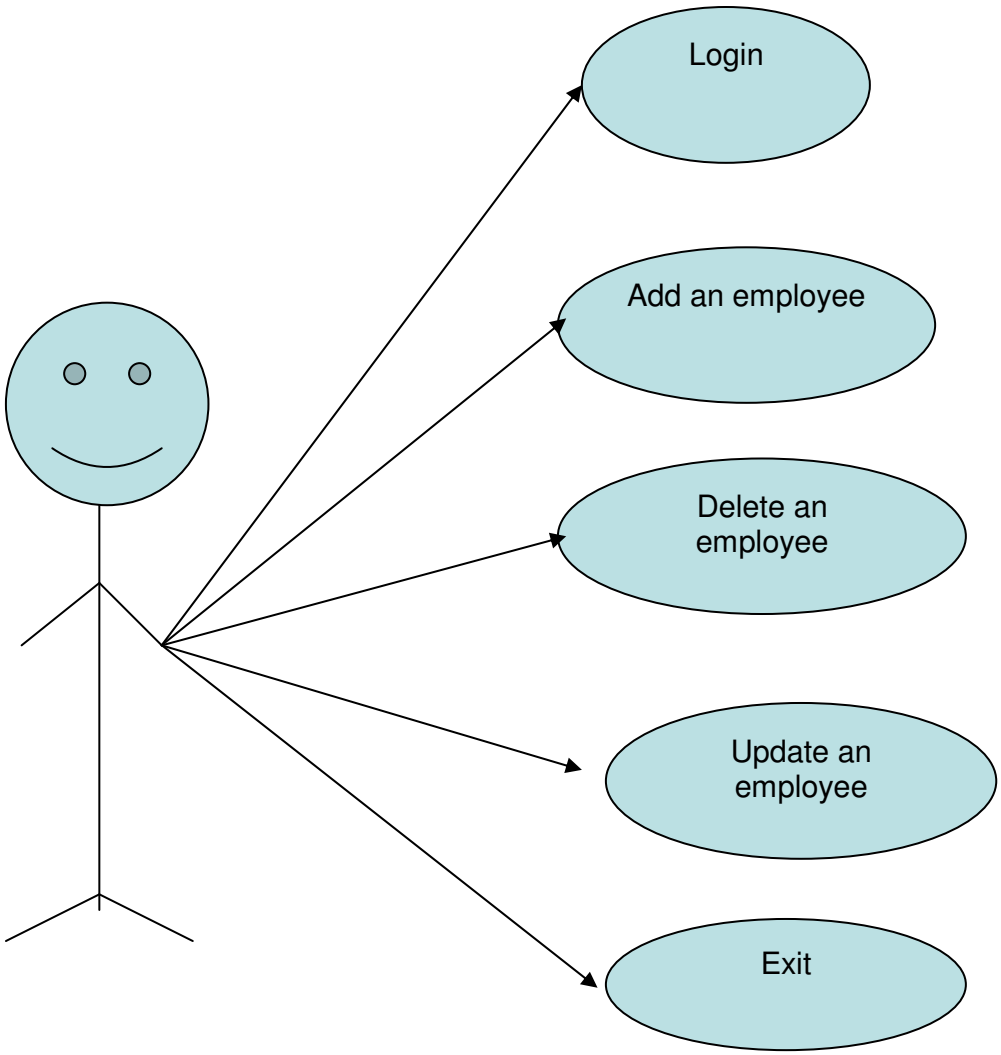


Figure 4 Use case for administrator

3.3.6.2 Fully dressed use cases for the administrator as the actor

Use case: Administrator login

Primary actor: Administrator

Stakeholders and interests:

Company: wants secure, reliable issue management system

Preconditions: The issue tracking system is prompting the user to enter username and password.

Success guarantee (post condition): The administrators has access to add, delete, update employees and also assign newly opened tickets to the technicians.

Main success scenario:

1. User entered username and password.

2. Presses the submit button.

3. If username and password is valid, the administrator login page is displayed.

Extensions (alternate flow)

1. If username or password is invalid, the error page is displayed and user is prompted to enter the right username and password.

Use case: Add an employee

Primary actor: Administrator

Stakeholders and interests:

Company: wants secure, reliable issue management system

Preconditions: The issue tracking system is prompting the user to enter username and password.

Success guarantee (post condition): The administrator fills out the new employee form and adds a new employee to the database.

Main success scenario:

1. User entered username and password.

2. Presses the submit button.

3. If username and password is valid, the administrator login page is displayed.

4. The login page as 5 options, one of which is add employee, on selecting the add employee button, the new employee form is displayed.

5. The user needs to fill all fields appropriately and hit the submit button.

6. A success message is displayed and the new employee is added into the database.

Extensions (alternate flow)

1. If all the required fields are not filled, the error page is displayed and user is prompted to fill all required fields.

2. If the application was unable to add the new employee information into the database due to some system error, an error message is displayed and the user has the option of going back to the administrator home page and tries to enter information again.

Use case: Add an employee

Primary actor: Administrator

Stakeholders and interests:

Company: wants secure, reliable issue management system

Preconditions: The issue tracking system is prompting the user to enter username and password.

Success guarantee (post condition): The administrator fills out the new employee form and adds a new employee to the database.

Main success scenario:

1. User entered username and password.

2. Presses the submit button.

3. If username and password is valid, the administrator login page is displayed.

4. The login page as 5 options, one of which is add employee, on selecting the add employee button, the new employee form is displayed.

5. The user needs to fill all fields appropriately and hit the add employee button.

6. A success message is displayed and the new employee is added into the database.

Extensions (alternate flow)

1. If all the required fields are not filled, the error message is displayed and user is prompted to fill all required fields.

2. If the application was unable to add the new employee information into the database due to some system error, an error message is displayed and the user has the option of going back to the administrator home page and tries to enter information again.

Use case: Delete an employee

Primary actor: Administrator

Stakeholders and interests:

Company: wants secure, reliable issue management system

Preconditions: The issue tracking system is prompting the user to enter username and password.

Success guarantee (post condition): The administrator deletes the selected employee from the database.

Main success scenario:

1. User entered username and password.

2. Presses the submit button.

3. If username and password is valid, the administrator login page is displayed.

4. The login page as 5 options, one of which is delete employee, on selecting the delete employee button, a table with all employees is displayed.

5. The user chooses the employee he wants to delete by using the radio button and hits the delete button.

6. A success message is displayed and the selected employee is deleted from the database.

Extensions (alternate flow)

1. If the application was unable to add the new employee information into the database due to some system error, an error message is displayed and the user has the option of going back to the administrator home page and tries to enter information again.

Use case: Update an employee

Primary actor: Administrator

Stakeholders and interests:

Company: wants secure, reliable issue management system

Preconditions: The issue tracking system is prompting the user to enter username and password.

Success guarantee (post condition): The administrator is able to update the username and password for a selected employee.

Main success scenario:

1. User entered username and password.

2. Presses the submit button.

3. If username and password is valid, the administrator login page is displayed.

4. The login page as 5 options, one of which is update employee, on selecting the update employee button, a table with all employees is displayed.

5. The user chooses the employee he wants to update by using the radio button and hits the update button.

6. Selected employee information is displayed, and then the user hits the update button.

7. User has to now enter the new username and password and hit the update button.

8. A success message is displayed and the selected employee username and password is updated and stored into the database.

Extensions (alternate flow)

1. If the application was unable to update the employee information into the database due to some system error, an error message is displayed and the user has the option of going back to the administrator home page and tries to enter information again.

2. If the username or password is not filled, then an error message is displayed and the user is prompted to enter those fields.

Use case: Administrator assigns technician

Primary actor: Administrator

Stakeholders and interests:

Company: wants secure, reliable issue management system

Preconditions: Assign technician button is displayed on the administrator login page.

Success Guarantee (post condition): The administrator has successfully assigned a new technician to the newly entered ticket.

Main success scenario:

1. User logs in to the administrator login page.
2. User selects the assign ticket button, which then displays all the new tickets that are present at that time in a table.
3. User selects one of the displayed tickets based on priority , using a radio button which in turn displays the ticket information.
4. User then presses the update button, which then takes the user to the update page where the user can enter the name of the technician who needs to work on the ticket to resolve the problem.
5. User then presses the submit button and the ticket is updated to hold the newly assigned technician for that ticket and a success message is displayed.

Extensions (alternate flow)

1. If the application was unable to update the new technician information into the database due to some system error, an error message is displayed and the user has the option of going back to the administrator's home page and can try to enter information again.

Use case: Administrator exit

Primary actor: Administrator

Stakeholders and interests:

Company: wants secure, reliable issue management system

Preconditions: Exit button is displayed on the login page.

Success Guarantee (post condition): The administrator has successfully logged out of the system.

Main success scenario:

1. User hits the exit button.

2. User has exited the application. Logged out message is displayed.

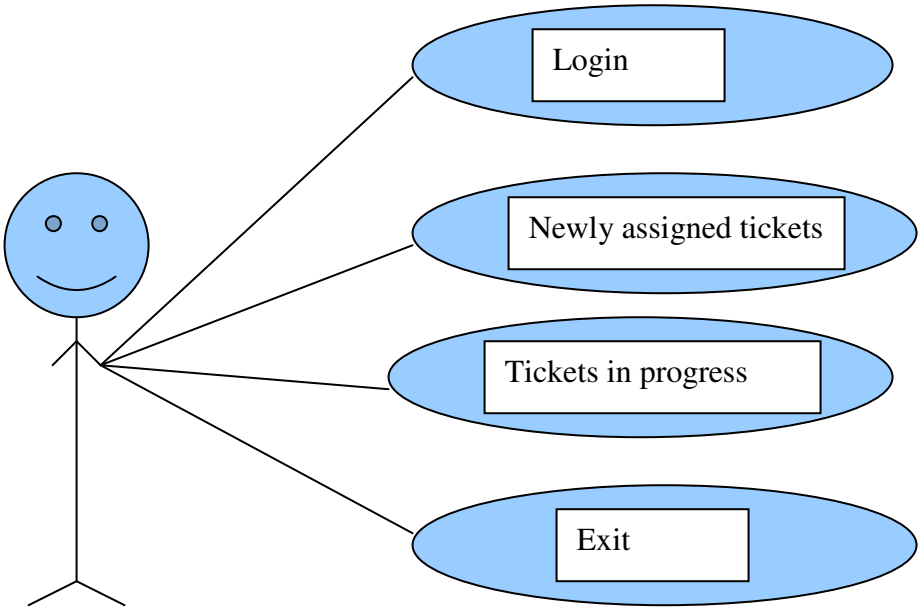


Figure 5 Use case for technician

3.3.6.3 Fully dressed use cases for the technician as the actor

Use case: Technician login

Primary actor: Technician

Stakeholders and interests:

Company: wants secure, reliable issue management system

Preconditions: The issue tracking system is prompting the user to enter username and password.

Success guarantee (post condition): The technician has access to tickets newly assigned to him and to update any other tickets that are in progress.

Main success scenario:

1. User enters username and password.

2. Presses the submit button.

3. If username and password is valid, the technician login page is displayed.

Extensions (alternate flow)

1. If username or password is invalid, the error page is displayed and user is prompted to enter the right username and password.

Use case: Technician to process newly assigned tickets

Primary actor: Technician

Stakeholders and interests:

Company: wants secure, reliable issue management system

Preconditions: The issue tracking system is prompting the user to enter username and password.

Success guarantee (post condition): The technician gets a list of the tickets that are in assigned state, he tries to fix the problem based on available information and puts the ticket into the to be verified state or if he needs more information, he puts the ticket into the waiting state and waits for the call center agent to call the customer and get the needed information to fix the problem.

Main success scenario:

1. Technician enters username and password.

2. Presses the submit button.

3. If username and password is valid, the call center agent login page is displayed, which has 3 button options, one of which is newly assigned tickets.

4. Then a table is displayed with a summary of all tickets which are open and are in the assigned state, so that the problem can be solved by the technician.

5. User then can chooses one of the ticket by selecting a radio button and hits submit

6. Then a page is displayed with the existing ticket information for the selected ticket where the technician can review the existing information

7. The user then hits the Update button, which will take him to the update ticket form page.

9. User will then fix the problem and put ticket in to be verified state if all required information is present. User will put the ticket in the waiting state if he requires more information to fix the problem.

10. User hits the submit button.

11. Success message is displayed and the updated ticket is stored into the database and the user has option to go back to the technician login page by selecting the Go Home button.

Extensions (alternate flow)

1. If all required fields in the update ticket form are not filled, then an error message is displayed, and the user is prompted to fill all required fields.

2. If the application was unable to update the updated ticket information into the database due to some system error, an error message is displayed and the user has the option of going back to the technician home page and can try to enter information again.

3. If the state and progress fields are not filled appropriately, then an error message is displayed, and the user is prompted to fill them appropriately.

4. If there are no tickets in the assigned state, then a message is displayed and the user has an option to go back to the home page by hitting the go home button.

Use case: Technician to process tickets that are in the progress state

Primary actor: Technician

Stakeholders and interests:

Company: wants secure, reliable issue management system

Preconditions: The issue tracking system is prompting the user to enter username and password.

Success guarantee (post condition): The technician gets a list of the tickets that are in the progress state, he tries to fix the problem based on available information and puts the ticket into the to be verified state or if he needs more information, he puts the ticket into

the waiting state and waits for the call center agent to call the customer and get the needed information to fix the problem.

Main success scenario:

1. Technician enters username and password.

2. Presses the submit button.

3. If username and password is valid, the call center agent login page is displayed, which has 3 button options, one of which is progress tickets.

4. On selecting this button a table is displayed with a summary of all tickets which are open and are in the progress state is displayed.

5. User then can choose one of the ticket by selecting a radio button and hitting submit.

6. Then a page is displayed with the existing ticket information for the selected ticket where the technician can review the existing information

7. The user then hits the Update button, which will take him to the update ticket form page.

8. User will then fix the problem and put ticket in to be verified state if all required information is present. User will put the ticket in the waiting state if he requires more information to fix the problem.

9. User hits the submit button.

10. Success message is displayed and the updated ticket is stored into the database and the user has option to go back to the technician login page by selecting the Go Home button.

Extensions (alternate flow)

1. If all required fields in the update ticket form are not filled, then an error message is displayed, and the user is prompted to fill all required fields.

2. If the application was unable to update the updated ticket information into the database due to some system error, an error message is displayed and the user has the option of going back to the technician home page and can try to enter information again.

3. If the state and progress fields are not filled appropriately, then an error message is displayed, and the user is prompted to fill them appropriately.

4. If there are no tickets in the progress state, then a message is displayed and the user has an option to go back to the home page by hitting the go home button.

Use case: Technician exit

Primary actor: Technician

Stakeholders and interests:

Company: wants secure, reliable issue management system

Preconditions: Exit button is displayed on the login page.

Success guarantee (post condition): The technician has successfully logged out of the system.

Main success scenario:

1. User hits the exit button.

2. User has exited the application. Logged out message is displayed.

3.4. Database schema for the issue tracking system

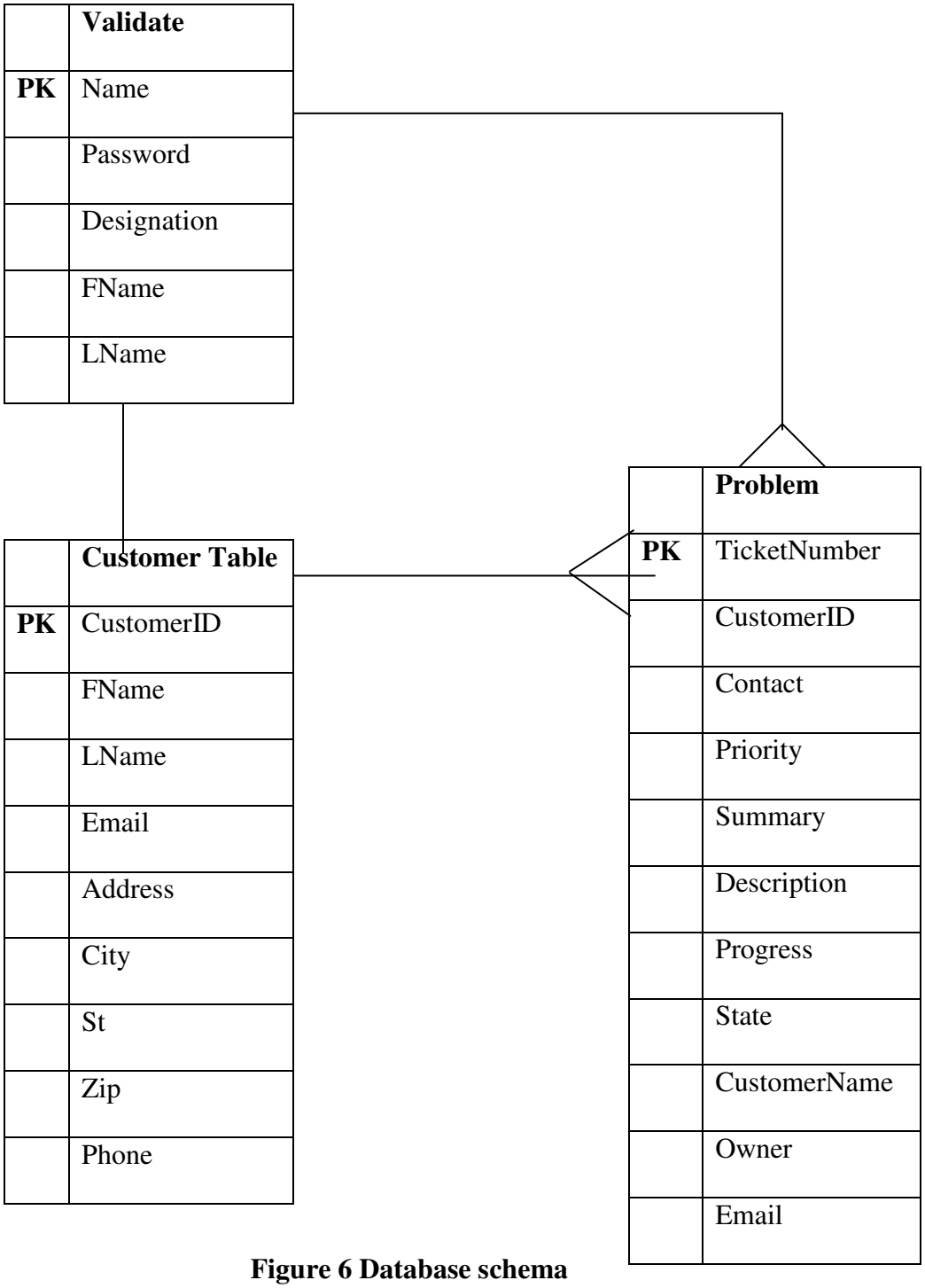


Figure 6 Database schema

The database used for the application is MySql and the tables in the database are show in the figure 6. The validate table holds the information to log on to the issue tracking system. It contains the name of the employee, his username and password needed for login and the designation which determines which view to display after the employee had logged in.

The customer table holds all data relevant to a customer like name, customerID, address and contact information. The problem table holds all the tickets information like the ticketnumber, the state of the ticket, the description of the problem, who the technician is working on the ticket and the customer to whom the ticket belongs to.

3.5 Snap shots

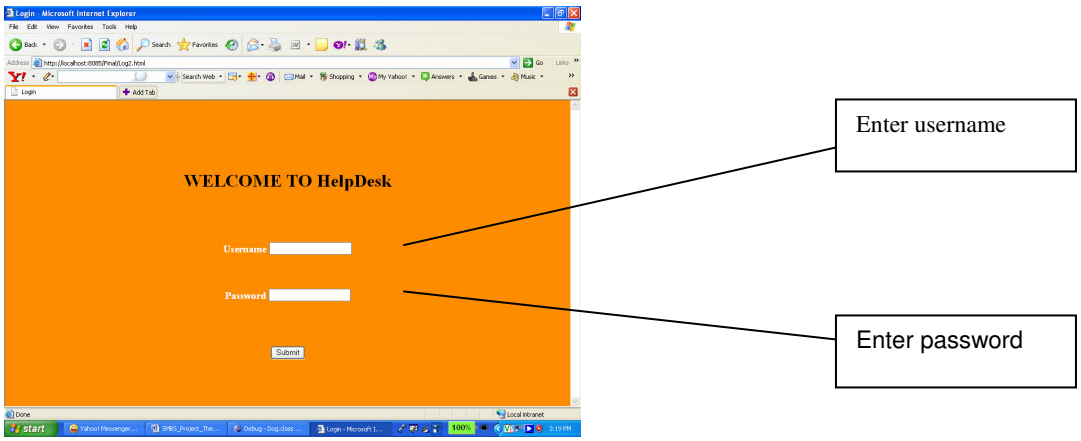


Figure 7 login page

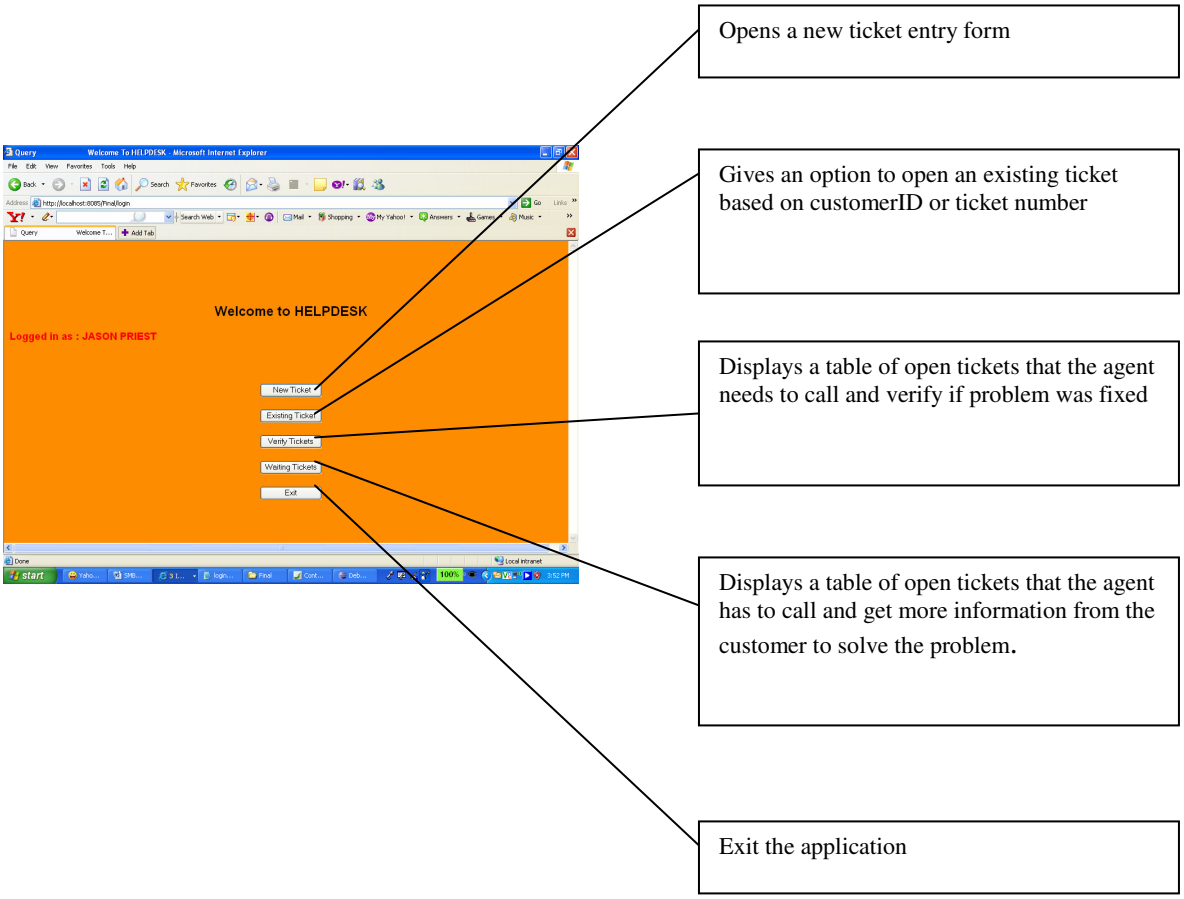
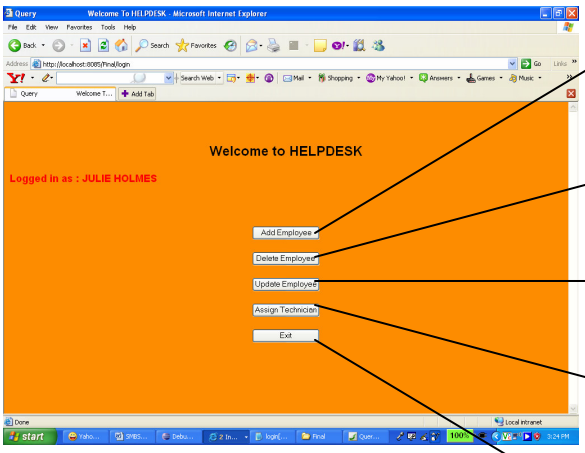


Figure 8 Call center agent login page



Opens an add employee form

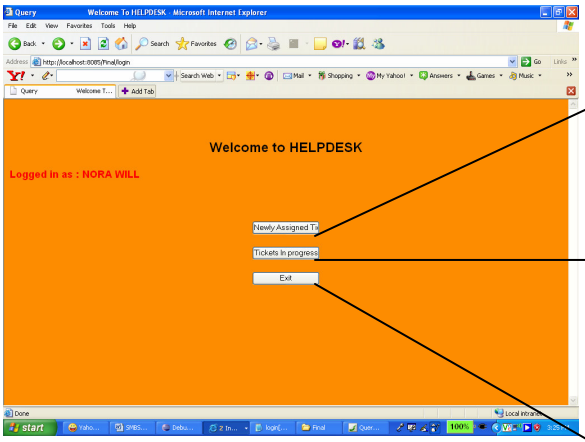
Displays a table of employees, and one can be selected to be deleted.

Displays an update form to enter the new username and password.

Displays a table of all new tickets that need to be assigned to a technician.

Exit the application

Figure 9 Administrator login page



Displays all newly assigned tickets

Displays all tickets that are in the progress state.

Exits the application

Figure 10 Technician login page

3.6 Review of deliverables

- High level design in form on workflow diagram and use case diagrams.
- Detailed design in form of fully dressed use case description.
- User guide in form of labeled screen shots.
- Tested and functional code executables.

Resource requirements

Customer service agent workstation requirements

Hardware component	Microsoft Windows
Hardware Architecture	IBM Compatible
CPU	500 MHz
RAM	256 MB
Free Disk Space	400 MB
Load Device	N/A
Recommended Monitor Resolution	1280x1024 (256 colors)
Minimum Monitor Resolution	1024x768 (256 colors)

Table 2 Customer Service Agent Workstation Requirements

Server Machine

- A 1.0 GHz Intel Pentium III-compatible (or faster) processor is required. A 2.0 GHz (or faster) Intel Pentium 4-compatible dual core processor or 2.8 GHz (or faster) Intel Pentium 4-compatible single core processor is recommended.
- A minimum of 512 MB of RAM is required. 1 GB of RAM is recommended.

Operating System

- Windows 2000 Server, or
- Windows Server 2003 (32-bit version)

Web Server

- Tom Cat version 5.1

Database

- MySQL 5.0

Software to be deployed

- Small business ticketing system software application executables.

3.7 Outcomes

The collective outcome of the project is a fully functional, tested and documented issue tracking and ticketing software solution that can be deployed on a server, connected to a database and accessed using a web browser. The testing performed includes unit, system and integration testing. A list of references for all research performed is also presented as part of the documentation.

4.0 Chapter 4

4.1 How the project began

In the duration of the master's program, a lot of Java technologies were learned in theory and this master's project is the avenue to implement these technologies in order to get first hand experience. The call center industry is really popular and a lot of resources were available to provide guidance on a ticketing system. Also from personal experience, a local service provider for cable Internet and TV cable service exists in the Broomfield area of Colorado. This provider does not have a ticketing system and a lot of problems are faced because of the lack of customer support software. This sparked an idea of developing a ticketing system for a small business.

Once the requirements were finalized, the project timeline shown below was put together. In order to meet the timeline, some activities had to be done in parallel. The modules that had functionality that were common to each other were initiated in parallel. The debugging required had to be done just once and re-used across the common modules. Unit testing of the modules was also done as the project progressed as opposed to waiting till the end to perform testing. A few challenges were faced along the way e.g. identifying an IDE to develop the web page front end. Microsoft web expressions were used to design the pages. It helped in speeding up the process of creating and designing the web pages so that more time could be spent on developing the functionality of the design itself.

4.2 How the project was managed

The project had a set project plan as shown below with specific milestones and deliverables due on specific dates. The project was for the most part individual based and hence management was pretty easy.

Project task plan and milestones

ID	Task Name	Duration	Start	Finish	Resource Names	% Complete
1	Master Thesis and Prof Project	104 days	Mon 10/23/06	Thu 3/1/07	Mohita	100%
2	Proposal Statement	7 days	Mon 10/23/06	Mon 10/30/06	Mohita	100%
3	Thesis Statement	7 days	Mon 10/30/06	Tue 11/7/06	Mohita	100%
4	Project Proposal	5 days	Mon 11/6/06	Fri 11/10/06	Mohita	100%
5	Finalize Project proposal	3 days	Tue 11/14/06	Thu 11/16/06	Mohita	100%

6	Project advisor selection	2 days	Fri 11/17/06	Mon 11/20/06	Mohita	100%
7	Project Proposal Approval	1 day	Mon 11/20/06	Mon 11/20/06	Project Advisor	010%
8	Requirements Gathering	3 days	Mon 11/20/06	Wed 11/22/06	Mohita	100%
9	Requirements Specification	2 days	Fri 11/24/06	Sun 11/26/06	Mohita	100%
10	High level design	7 days	Sun 11/26/06	Sun 12/3/06	Mohita	100%
11	High level design approval	3 days	Sun 12/3/06	Tue 12/5/06	Mohita	100%
12	Detailed Design	5 days	Tue 12/5/06	Sun 12/10/06	Mohita	100%
13	Detailed Design Approval	3 days	Sun 12/10/06	Tue 12/12/06	Mohita	100%

15	Coding	28 days	Mon12/13/06	Sun 1/28/07	Mohita	100%
16	Unit Testing	3 days	Mon 1/29/07	Wed 1/31/07	Mohita	100%
17	System Testing	4 days	Wed 1/31/07	Sun 2/4/07	Mohita	100%
18	Advisor approval of final executables	3 days	Sun 2/4/07	Tue 2/6/07	Mohita	100%

Table 3 Project Task Plan

4.3 Changes to the project plan

The majority of the changes to the project plan occurred to the front end user interfaces. Initially a lot of thought went into planning the layout of each user interface page ,so that the application could look professional , but on a later thought the look and feel of the page was compromised, so that the functionality of the application could be developed and delivered on time. The changes were aimed at making usage easier. Due to the limited amount of time the customer view was deleted during the development of the project which could be added at any later time. Some of the functionalities such as display of the date and current time on every page and validation of the input from the user were left out due to time constraints. Also in certain cases, when challenges were

incurred in debugging specific modules, the plan slipped in order to ensure the module in question was debugged and worked as per specification. This time was made up by adding man hours to the plan by working late in order to stay on track with the timeline. In some instances, a lot of progress was made on the weekends in order to stay on track and gain time.

4.4 Evaluation of whether project met goals

The project's initial goal was to design functionality as explained in the use case. The idea was to develop a GUI and a backend database that would work together to provide the ticketing system functionality. Based on this, the project met all goals defined at the start of the project with certain modifications made to the requirements. The design put together was met. The final project was submitted for approval. Since the software will not be deployed in any production environment, it is hard to determine the extent of success it would meet in a live environment with real users using the software. Extensive testing performed on the software however indicates that if deployed in a small business environment, it would succeed in the intent for which it was designed for. Based on this and the approval of the project committee, the project is deemed a success. Further more, since the original intent was to gain hands on knowledge, a lot of technology and best practices in developing such an application was gained.

4.5 What went right and what went wrong

For the most part it was a great learning experience. To start with, the original software chosen to do the web page and front end design was Adobe Dream weaver, a lot of

difficulty was faced in trying to format the form elements and it was hard to perform many basic tasks. It was due to lack of experience in using this software that caused this as a certain level of experience or training is necessary to use this software. After that, Microsoft web expressions was chosen and this software was a lot more user friendly and required very little training and expertise for first time users. Also in the initial stages of the projects, separate servlets for each web page were being used but later it was discovered that servlets could be merged and the code written for other web page could be re-used.

For “the what went right part”, many features that were designed for some of the initial modules and took a lot of time were re-used in the other modules and extended to create specific functionality for that module. The codes re-use approach, which is a best practice for software development saved time on the project plan.

4.6 Findings/Analysis of results

For the most part it has been a very good learning experience. Given the knowledge and tools used, the project is deployable in a small service provider business for a ticketing system. The solution has not been load or stress tested but in the unit, system and integration testing, all modules and options work well. User front end, middle tier functionality and back end functionality work well together. Feedback was solicited from an industry expert and given the boundaries of the design, all the feedback received was favorable.

Further research indicated that for this software to be deployed in a commercial environment, in addition to the hardware and software specifications listed above, it was

identified that for an operation to be able to obtain full redundancy and failover in order to provide high reliability, a few software modifications as well as extra hardware would be needed. The program was designed and developed with the assumption that downtime would be incurred in case of problems with the software. Full redundancy was not considered.

5.0 Chapter 5

5.1 What was learnt from the project experience

In the course of the project, experience was gained on the overall life cycle development of a project. It was like developing a project in real time for a commercial application. Experience was gained in requirements gathering, analysis, design, coding, unit testing, integration testing and regression testing. All these concepts were theoretical as learned and this was a chance to implement the concepts.

Experience was also gained in putting a project plan together in MS Project and sticking to the plan itself. Some other experiences gained were working with Microsoft web expressions for web design. During the course of the project, a lot of challenges were faced in the implementation phase. Challenges like errors encountered and changes in implementation strategy to overcome these errors in order to maintain the timeline.

Also on the documentation front, APA formatting was mastered while preparing the master's thesis. A lot of aspects of the thesis were new ideas and a learning experience. Finally, the goal of the master's program on the whole was to become an expert with Java and related technologies. This project provided that experience to learn Java, servlets, database technologies, web design and the overall joint working of these technologies to make this project work.

5.2 What could have been done differently

The following items could have been implemented slightly differently so as to improve the application.

Use of Apache Struts instead of using Java servlets as the controller. Struts provide a unified framework for deploying servlet and jsp applications that use the MVC architecture. Instead of having several servlet controllers we can have just one controller using Struts. Struts uses xml files to determine which servlet needs to service a particular http request.

Another reason for using Struts would be the built-in capabilities for checking that form values are in the required format. If values are missing or in an improper format, the form can be automatically redisplayed with error messages and with the previously entered values maintained. This validation can be performed on the server (in Java), or both on the server and on the client.

Scripting has been used in JSP which means that there is java code present in the jsp pages. In the future if someone without experience in java has to read or change the code in the jsp would find it a daunting task. By using JSTL(Java Standard Tag Library), custom tags (tags written by users) and EL(Expression Language) it is possible to limit the usage of java in jsp making it more readable for users who have no java experience..

In addition to the above points, if given an opportunity, a real-life project with a few team members and real-life requirements would have been a richer experience. Although the requirements of this project were real-life based, the final product was never going to be deployed in a live environment. A few more challenges would have presented themselves in a live deployment and a more thorough testing would have dug up a few more bugs and made the program efficient.

5.3 Improvements/Evolution of the project

Given more time and resources, the following improvements could be made to the project:

1. A customer view could be implemented, so that the customer is able to change customer related information like change in address, phone number, email etc.
This view could also be used to check the current status on all the open tickets for that particular customer.

2. Administrator functionalities could be increased to solve some of the very generic problems instead of assigning all tickets to the technician.

3. While assigning tickets, displaying tables of existing tickets, sort functionality could be added, so that tickets could be sorted by priority, date etc.

4. The ticketing system could be modified to include functionality to check for duplicate tickets that are opened for the same customer and same issue instance.

5. The ticketing system could also be enhanced to maintain history of the call center agents handled, priority of tickets etc. Currently only the description history is maintained for the duration of the ticket. For all other fields only the last occurrence is maintained.

6. The GUI design can be more efficiently designed for logical usage. It has been designed to support the design shown and not a lot of focus has been given to the flow efficiency of the GUI.

7. Extend the ticketing system to cater to the email channel of problem resolution.

5.4 Conclusions

As stated earlier, it was a really good learning experience to implement this project. It started off as a student project to learn and get hands on experience on the technologies learned in class. The design started coming together and at the end of the project, functionality built in the project could be actually used in a small services business. The project has been tested on the Windows OS and MySQL database but if this project was done at a commercial level, a lot more platform testing would have been done. The requirements were solicited from an industry expert and hence make it a real life project. This project although designed with real life requirements in mind will probably never be deployed in a real live production environment.

References

Bridge pattern (2007, January).Wikipedia, the free encyclopedia. Retrieved January 2, 2007 from http://en.wikipedia.org/wiki/Bridge_pattern

Commercially available ticketing system 1 (n.d). Retrieved February 17, 2007 from http://www.salesforce.com/products/service-support/

Commercially available ticketing system 2 (n.d). Retrieved February 17, 2007 from http://www.microsoft.com/dynamics/crm/product/overview.mspx

Commercially available ticketing system 3 (n.d). Retrieved February 17, 2007 from http://www.entellium.com/smallbiz.html

Glossary 1 (n.d). Sun development network. Retrieved January 5, 2007 from http://java.sun.com/products/jsp/jstl/

Glossary 2 (n.d). J2EE Tutorial. Retrieved January 5, 2007 from http://java.sun.com/j2ee/1.4/docs/tutorial/doc/JSPIntro7.html

Glossary 3 (2007, January). Wikipedia, the free encyclopedia. Retrieved January 7, 2007 from http://en.wikipedia.org/wiki/Customer_Relationship_Management

Issue tracking system comparison (2007, February). Wikipedia, the free encyclopedia Retrieved February 2, 2007 from http://en.wikipedia.org/wiki/Comparison_of_issue_tracking_systems

Java design pattern (2007, January). About design pattern. Retrieved January 8, 2007 from http://www.javacamp.org/designPattern/

Pisarski.S.M (2005, December). Conference website redesign. Retrieved January 10, 2007 from http://165.236.235.140/lib/SPisarskiPartI2005.pdf

Roetker.M (2006, February). Scouting eCommerce. Retrieved January 10, 2007 from http://165.236.235.140/lib/MRoetkerPartI2006.pdf

Trouble ticket (n.d). SearchCRM. Retrieved January 12, 2007 from http://searchcrm.techtarget.com/sDefinition/0,,sid11_gci927714,00.html

Use case diagram (n.d). Developer. Retrieved January 7, 2007 from http://www.developer.com/design/article.php/2109801

Workflow scenario (n.d). Retrieved January 8, 2007 from http://distribution.cs.ncl.ac.uk/projects/workflow_system/98-03-10.pdf

Appendixes

A. Definition of terms

Application Programmers Interface (API) – The means that an application/program accesses the operating system or other services at the source code level.

Browser – A program that accesses and displays files and other data available on the Internet and other networks.

Client – A computer program that can invoke other programs.

Hyper Text Transfer Protocol (HTTP) – The protocol used to transmit files, web pages, and web components over the Internet or other network connections.

Hyper Text Markup Language (HTML) – A text-based language used by browsers to render graphical content generally delivered by the Internet.

Interface – The point at which communications are initiated with a computer program and an entity such as another computer program, human, or peripheral.

Java Server Pages (JSP) – A language that combines HTML and Java technologies and allows for dynamic web page creation. JSP pages must be served up using a standard

servlet container such as Tomcat.

Java Virtual Machine (JVM) – A program layer that resides above the operating system level that runs Java byte-code and is the key feature of the Java programming language which makes it platform independent.

Model-View-Controller (MVC) – A pattern for designing an interactive application. The model denotes the internal workings or application's services. The view and controller refers to how the user sees the state of the model and changes its state by providing input respectively.

MySQL – A popular open source database. (http://www.mysql.com/)

Object-orientated programming– A programming paradigm based on the concept of encapsulating a data structure with associated actions called methods.

Open source – Software that is distributed free of distribution restrictions.
This does not necessarily mean that the software is free of charge, as there are a number of different open source licenses available.

Prototype – A partially working model of the user-interface that the customer can access for user acceptance testing during the development phase of developing an application.

Rapid Application Development (RAD) – An iterative software development methodology, which attempts to quickly deliver prototypes during the development stage to the customer for testing. The goal of this methodology is to deliver an application that fully meets the customer's needs quickly with minimal defects.

Relational database – A database that contains multiple tables consisting of rows (records) and columns (fields), which can be related to each other.

Servlet – A Java program that runs inside of a web server and responds to HTTP requests from clients.

Structured Query Language (SQL) – A standard database language utilized for creating, updating, or querying a relational database.

Customer Relationship Management (CRM) - is a broad term that covers concepts used by companies to manage their relationships with customers, including the capture, storage and analysis of customer information

Issue Tracking System (ITS) - is a software application that allows an enterprise to record and follow the progress of every problem or "issue" that a computer system user identifies until the problem is resolved.

Struts – A flexible control framework based on Java standards such as

servlets, XML, and JavaBeans to help enforce the Model-View-Controller design paradigm. (http://jakarta.apache.org/struts/)

System test – An environment that mirrors the production environment as closely as possible utilized for testing an application prior to release.

Tomcat – A servlet container that is used to send and retrieve HTTP/HTTPS transmissions over a network connection. It is generally used for the generation of dynamic web pages using the Java Server Pages (JSP) technology.

Uniform Resource Locator (URL) – An address of a document or resource located on the Internet or an Intranet.

Unit test – The act of testing individual pieces of software by a developer during the development phase of an application.

Web server – A process running on a server to send and receive HTTP transmissions over the Internet or an Intranet when utilizing a URL.

A Request for Comments (RFC) - is a formal document from the Internet Engineering Task Force (IETF) that is the result of committee drafting and subsequent review by interested parties. Some RFCs are informational in nature. Of those that are intended to become Internet standards, the final version of the RFC becomes the standard and no further comments or changes are permitted. Change can occur, however, through

subsequent RFCs that supersede or elaborate on all or parts of previous RFCs.

Expression Language (EL) - An expression language makes it possible to easily access application data stored in JavaBeans components.

Java Standard Library Tags (JSTL)- encapsulates as simple tags the core functionality common to many Web applications. JSTL has support for common, structural tasks such as iteration and conditionals, tags for manipulating XML documents, internationalization tags, and SQL tags. It also provides a framework for integrating existing custom tags with JSTL tags.

B. Comparisons of some of the available issue tracking systems

System	Creator	License	Implementation language	User interface	Back end	Customizable workflow
Argus	Thomas Mango	BSD	Java	web	PostgreSQL	
CodeBeamer	Intland Software	proprietary	Java	web, e-mail, Web services, eclipse plug in, NetBeans plug in	MySQL, Oracle, Derby (Java DB)	Yes
Teamwork (software)	Open Lab	proprietary, GPL	Java	web, e-mail	All relational (Hibernate)	
VisionProject	Visionera AB	proprietary	Java	web (Hosted or installed version), e-mail	Microsoft SQL Server, MySQL	Yes
Web Help Desk	MacsDesign Studio, LLC	proprietary	Java	web, e-mail	OpenBase, FrontBase, MySQL, Microsoft SQL Server, PostgreSQL	

Track+	Trackplus	proprietary	Java	web, e-mail	Firebird, Hypersonic SQL, InterBase, Microsoft SQL Server, MySQL, Oracle, PostgreSQL	
FIT	Alcea Technologies, Inc.	proprietary	HTML/Java on all platforms	web, e-mail	flatfile or JDBC (MS SQL2000, Oracle, MySQL)	yes
Scarab	Tigris.org	Apache-style license	Java	web	MySQL, PostgreSQL, Oracle[3]	Yes [4]
StarTeam	Originally Starbase Corporation, currently Borland	proprietary	Java, C++, others	6 varieties of custom client (Java IDE, command line, integrated into various Borland IDEs, custom SDK clients, Visual Studio client)	Microsoft SQL Server recommended, Oracle, DB2 supported	