PROJECT REPORT ON

PAYROLL SYSTEM

BY

ASMAU SANI MOHAMMED (905017)

HAMMAN W. SAMUEL (905061)

MALACHY KHANOBA (905109)

OSAETIN EVBUOMA (905076)

SOFTWARE ENGINEERING I (SEN 261), SPRING 2007

SUBMITTED TO

PROFESSOR AUGUSTINE ODINMA

ABTI-AMERICAN UNIVERSITY OF NIGERIA

APRIL 2007

DEDICATED TO

MRS. SHAHIDA FLORENCE SAMUEL

ABSTRACT

Project Objectives

The Software engineering course that we took this semester placed emphasis on the paradigm of eXtreme Programming (XP) techniques. Extreme programming is a programming technique that bases its values on simplicity, communication, feedback and courage. It encourages team work and constant communication with the client. The objective of this project is to put into practice the teachings that we have learnt about XP.

Approach

When we were first given this project, we met to determine how we were to carry out the task assigned to us. We drew up a time-line, discussed about the programming language to use to carry out the task, how the GUI would look like and also to make sure that we understood what was assigned to us. We finally settled for Visual Basic (VB) as our programming language. We got more information on what we were to do and set about completing our task, making use of the new ideas taught in class, and especially spikes.

Achievement

Our task was to develop a payroll system that would keep a record of employee data including their pension plan, union membership status, and taxes and also to be able to calculate the pay of the employees taking into consideration employee data. We have been able to achieve these task. The software we developed calculates the employee net pay from the deductions. The payslip can be printed out as a receipt. Most of the bugs that we found and those that the clients and beta users found have been corrected. Any new bugs found will also be corrected and the software will be updated and released. Because we used object-oriented principles, modifying the software to fix bugs or add a new feature has been relatively easy.

TABLE OF CONTENTS

CHAPTER 1
PROJECT BACKGROUND

1.1 Definition of Problem

In XP, the user story serves the purpose of the problem statement and the specifications. The user story provided by the client was as follows:

> "You are to design and implement a payroll system that should accept employee hours worked, compute net pay and record all the payroll data for subsequent processing. The system should prepare pay cheques and a payroll ledger, and maintain data on a sequential payroll file. Non-statutory deductions such as union dues and pension plans to be made.
>
> "The payroll data are employee number, employee name, pay rate, and union member flag. The year-to-date total should contain earnings, federal tax, pension plan, and union dues"

To accomplish these tasks, we had to meet with the client to find out exactly what the program is meant to do. Here are the sub stories that we were able to get from the client:

- The program accepts employee hours worked
- The program computes net pay
- The program record all the payroll data for subsequent processing
- The program should prepare pay cheques
- The program should prepare a payroll ledger
- The program should maintain data on a sequential payroll file
- Non-statutory deductions such as union dues and pension plans to be made
- Year-to-date total should contain earnings, federal tax, pension plan, and union dues
- Payroll data are employee number, employee name, pay rate, and union member flag

1.2 Terminology Definition

This section gives a definition and explanation of some of the terms used in the project:

Employee Data: This is the employee's information in the company. It consists of the employee identification number, employee name, pay rate, pension plan flag, and union member flag.

Payroll Records: The payroll records are used to store each month's hours worked, and the rates for that month.

Rates: Rates consists of the percentage that would be deducted from the gross pay depending on union membership status, pension plan, state and federal tax. Each employee can have a unique hourly rate.

Payroll Ledger: This is a table that shows the calculated pay of employees and the month in which they earned the pay. The ledger can be filtered by name, identification number, year and month.

Hours Worked: This is the number of times that an employee work in a month. The hours worked is used to calculate the pay that an employee will receive for that month.

Net Pay: The net pay is the final salary amount that would be given to the employee after all the deductions are subtracted from the gross pay. The deductions include among others taxes, union member dues and pension plan.

Gross Pay: The gross pay is the amount that the employee earns before the deductions are subtracted.

Deductions: Deductions are made up of taxes, union membership dues, pension plan. They are subtracted from the gross pay to give the net pay which is the employee's final pay for the month.

Taxes: The taxes consists of the state dues and federal dues. A percentage of the employee's salary goes to state and country.

Union Membership Dues: This is meant for employee's that are union workers in the company. They get to pay a percentage for union dues. An employee can be a union member and later change status to be a non-union member.

Pension Plan: Employees that opted to use the pension plan of the company get to pay a particular percentage of their pay in preparation for their retirement.

Payslip: These are similar to pay cheques. They allow the employee to have his or her pay printed out on paper so that they can cash it.

Year-To-Date Total: The year-to-date total is the summation of all the previous earnings till the month before the current month.

1.3 Definition of Project Objectives

The purpose of this project is to put into practice what we have learnt so far in our software engineering class. We spent most of the semester studying extreme programming. With extreme programming, the client knows how far he or her software is coming and knows what to except at one particular time or the other. The software is built exactly the way the client wants it to be built.

1.4 Project Deliverables

We met with our client on a few occasions to show our progress so far. In the long run, showing the client part of the finished product in stages helped. There were a few things he brought to our understanding and we were able to change those things. If we had waited till we finished the coding, we would have found it really difficult to correct the errors we found in such a short time.

The graphical user interface was acceptable to the client at this stage even though he thought we could improve on it with a little more time. From our project deliverables, the client was able to bring to our understanding that the employees should have different pay rates and that it would be safe (for record purposes) to calculate the employee's year-to-data total salary earnings. These observations were later incorporated into the final deliverable.

1.5 Project Plan

At the beginning of the project, we scheduled meeting time for the group to discuss on the design and implementation of the software and what language to use in writing the software. We had several meetings to this effect. When then developed a time-line for the project–when we would be releasing the first version for scrutiny and the estimated time we thought we would use for refactoring. We also pondered on a suitable name to give the project. We came up with the name “Piccolo” but it was later changed to “Payroll System” after our meeting with our client.

The group was then divided into two pairs that would work on parts of the code. We kept in touch with each other and whenever we had difficulties, we asked each other questions. On some occasions, we had to pretend we were the customer so as to try to figure out some of the things that user would desire, such as the friendliness of the user interface and ease of navigation through the software. We also created a blog so that we could communicate with each other, and also so that our client could track our progress http://project2.wordpress.com

When we discovered that we were behind schedule, we met to create a new time-line. We were behind schedule mostly because of our different engagements in school work and other personal issues. Fortunately for the team, we were able to stick to the new time-line and the project was completed in due time.

After the major part of the code was completed, we went ahead to lace the code, that is, adding exception handlers which will make the software more robust. We avoided this at the beginning because we did not want to get confused with too many lines of code.

The version of the software that has been released can still be improved depending on user response. So far, we have succeeded in completing the desired goal of from the user stories given to us. The beta software has been released about three times and we have been able to get good ideas from the various users on how to improve the functionality, reliability and robustness of the software.

CHAPTER 2
DESIGN

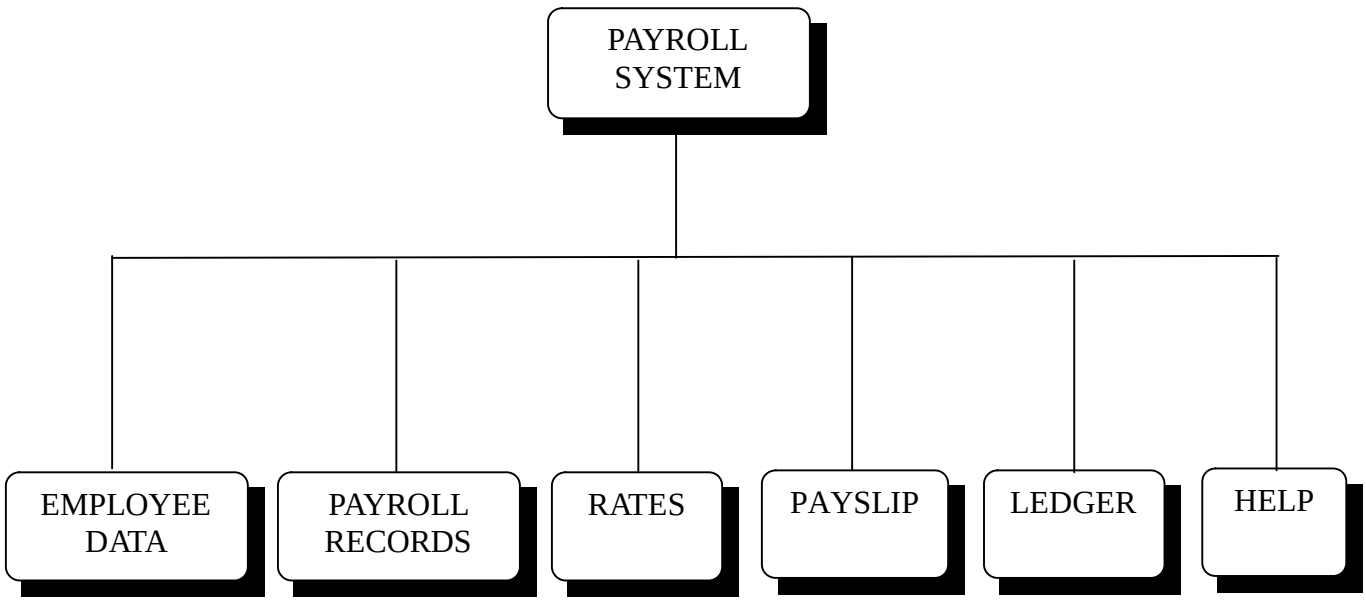
2.1 An Introduction to the methods and techniques adopted within the design

The software was built using Visual Basic and it can run on computers that have the .NET Framework installed. We divided the whole project into five main modules: employee data, payroll records, rates, ledger and help.

When a data is saved, the data gets written into an XML file. Keeping in line with making the software platform-independent, XML is also a highly platform-independent format for storing data. This is where all the data of the employee is stored, along with each month's payroll data for the employee. There is also a separate XML file for storing current rates. For an employee data or payroll data to be saved to the file, it has to be serialized. Getting any data from the file will require the data to first be deserialized before it can be used. Serialization and deserialization helped make the data storage very simple.

We also used the object-oriented paradigm to code and organize our program. This ensured code reuse, and efficient updating of the code later on.

2.2 High-Level Architecture Diagram of the main components



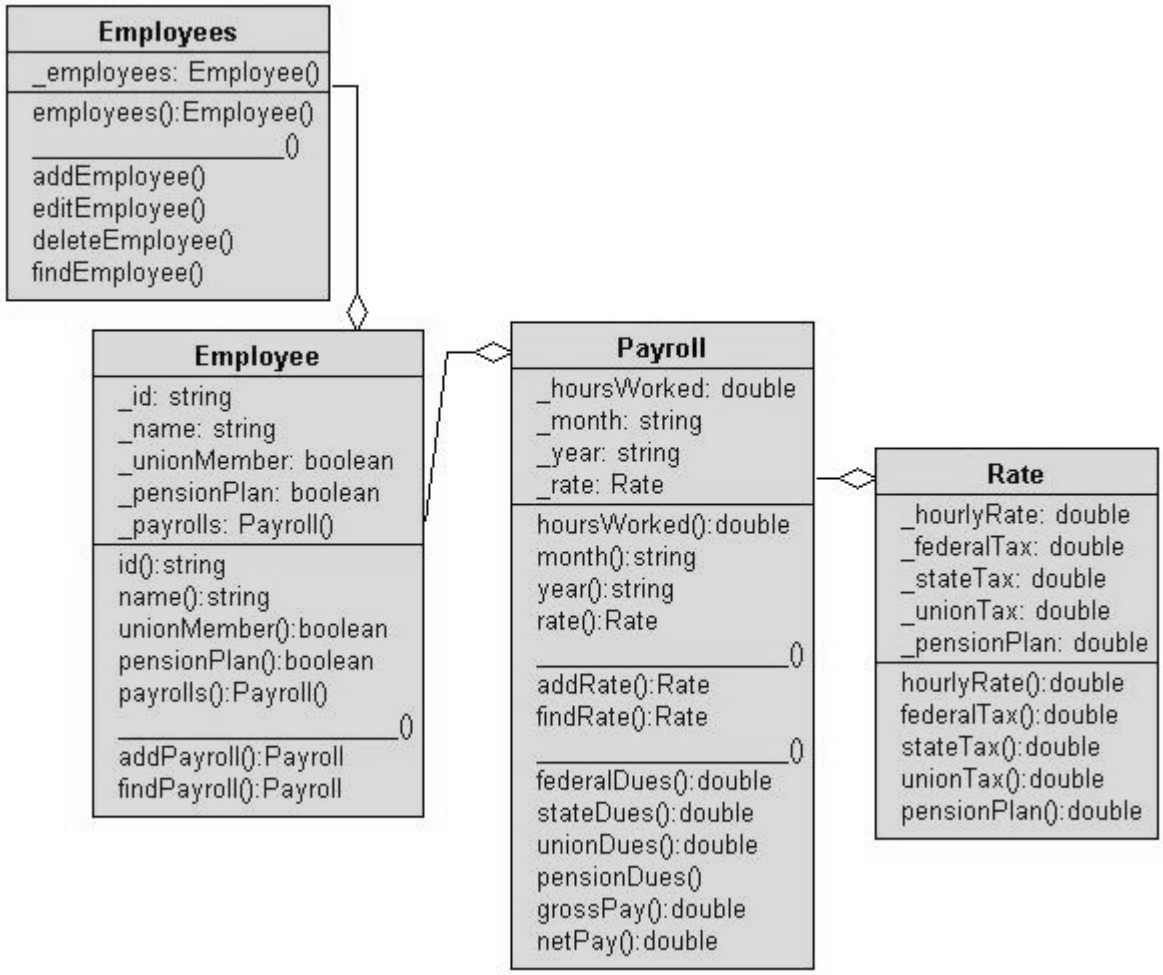
2.3 More detailed description of the most important modules of the design

The most important modules of the design are the employee data, payroll records and rates. For the employee data module, we had to keep track of the identification number, name, union membership status, pension plan agreement and payroll of an employee. We also had to link these to a general

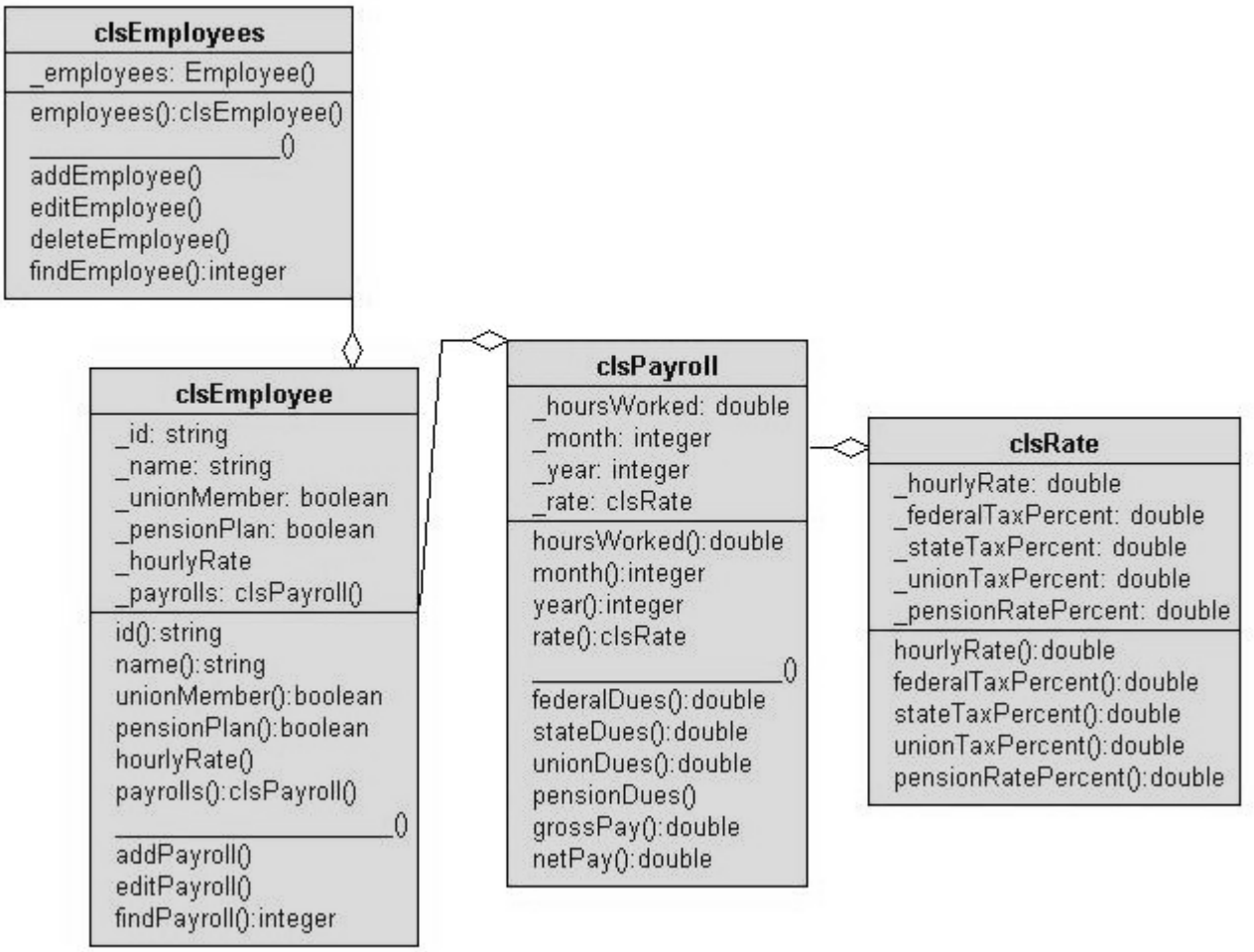
module that will hold all the other employees. There had to be a means of storing all these data, retrieving them, adding data and deleting data without giving any errors. From the employee data module, we had to create a link to the employee's pay records.

The payroll records had to contain the number of hours that the employee had worked and in what month and year that the employee worked. We also had to keep in mind the rate with which we would be able to calculate the employee's pay. The payroll records also had to contain the employee's deductions, that is, the deductions from the pension plan, union membership fee and taxes. Obviously, the payroll records also had to contain the gross and net pay of the employee.

The picture below shows a brief look at the modules of the original design.



We refactored the modules. This had to be done after one first deliverable was given to out client.



2.4 Indication of how the design fulfills the requirements

From the design picture above, it is clear that the software is meant to accept employee data, i.e., the identification number, name, union membership status and pension plan. It should stores these data in a XML file. The software computes the employee payroll depending on the rate and deductions that has been specified for the employee. New employees can always be added to the list of existing employees. There is no limit to the number of employees that the list holds. An employee's data can be modified as well as deleted from the list.

Each employee has a link to a pay data in the payroll records module. From the design diagram, we can see that a new payroll data can be added to the employee's list of existing pay data. The payroll calculates the employee's pay depending on the hours worked, the employee's rates and deductions that include pension dues, union membership dues and taxes. Each pay data is stored according to the month and year is was calculated.

CHAPTER 3
IMPLEMETATION

Some of the difficulty we faced was in trying to make the desired functions work. At first the coding seemed a little complex, so we went back to the client for clarification and we were able to get a more detailed description of the functions. After studying the IDE we wanted to use, the coding became less complex and we were able to figure out how to go about it.

3.1 Technical issues encountered

Some of the technical issues and questions we encountered were with the following:

- How the program saves data
- How errors are detected and the error messages that should appear
- What happens when we try overwriting data that already exists on the list
- Is the software user friendly?
- Is the Graphic User Interface elegant ?
- The platform the software should run on

We were able to answer and resolve these issues after meetings and discussions with the entire group.

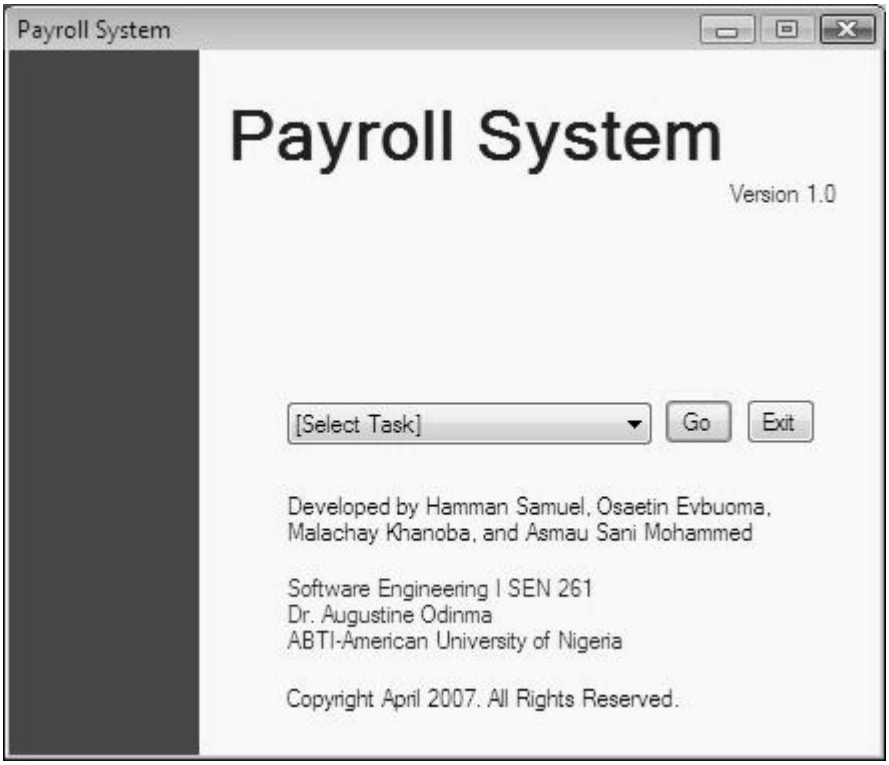
3.2 Software development practices adopted

- We used Extreme Programming to build the software.
- We adopted the **whole team approach** in which the client was involved through out the software development.
- We used **simple designs** which were as simple as the current functionality allowed by the system. We first began with very simple design which is incrementally improved.
- **Continuous integration** was also used where each code was compiled run and tested before adding to the system after which the system must meet all the criteria
- The most important practice adopted was **pair programming**. We paired up with members of the team and each team met at various times to code. After each pair coded, they made sure they tested and compiled it, then added it to the main system which they sent to the other group members.
- We also made sure that we coded according to **coding standards**, the development codes were consistent and had the same style through out.

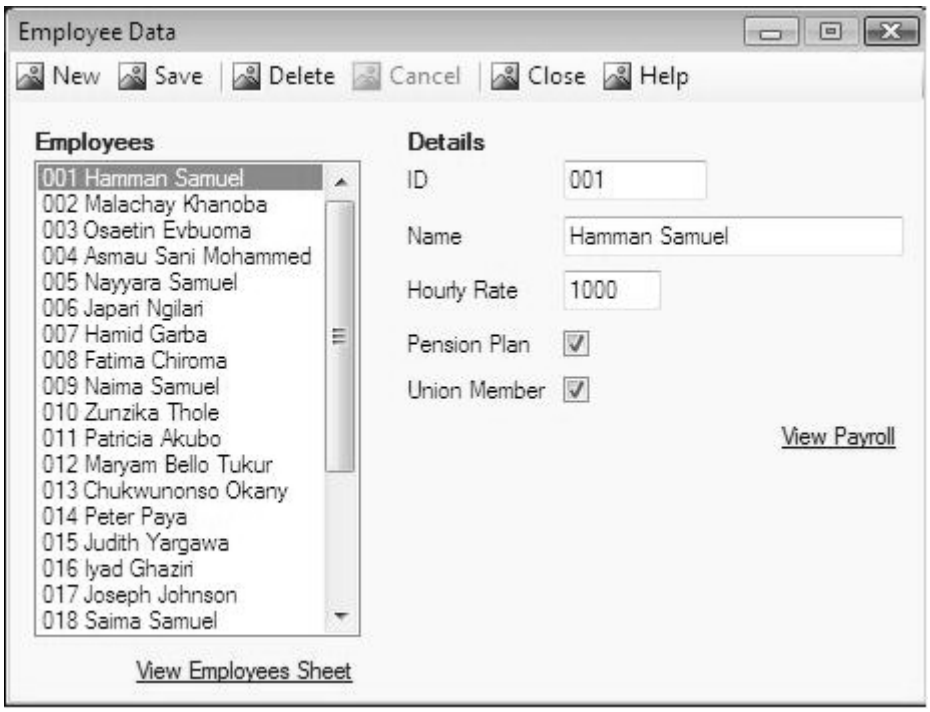
3.3 Screenshots of the final system

Below are some screenshots of the final program.

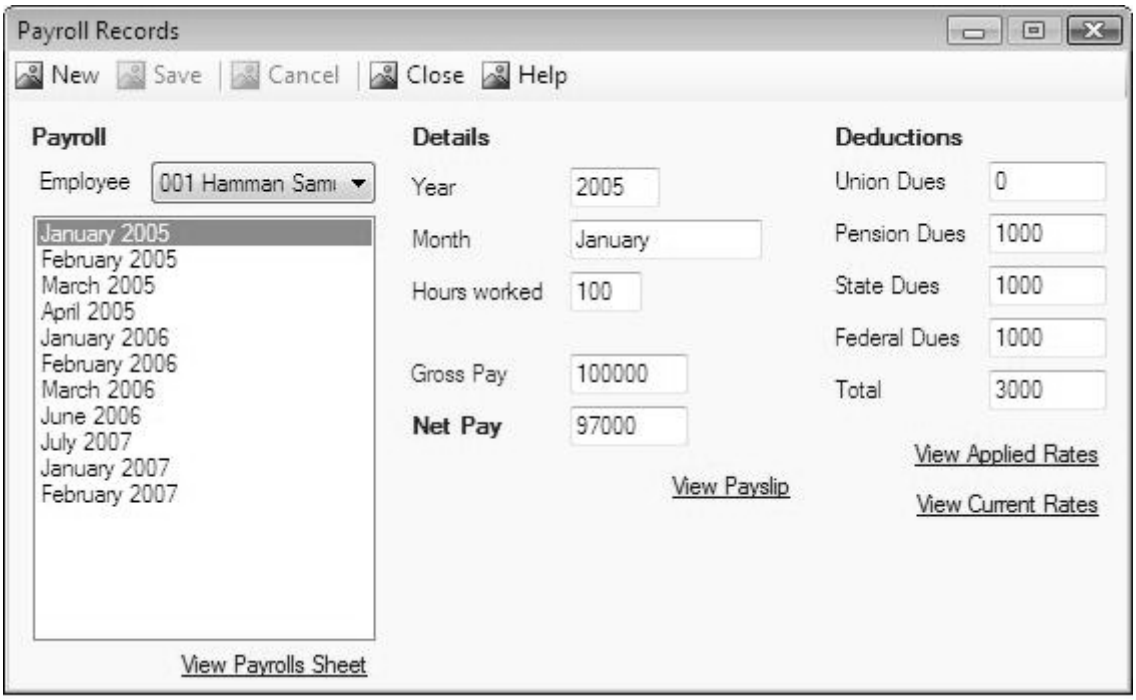
The main screen shows all the tasks that can be selected and performed by the user.



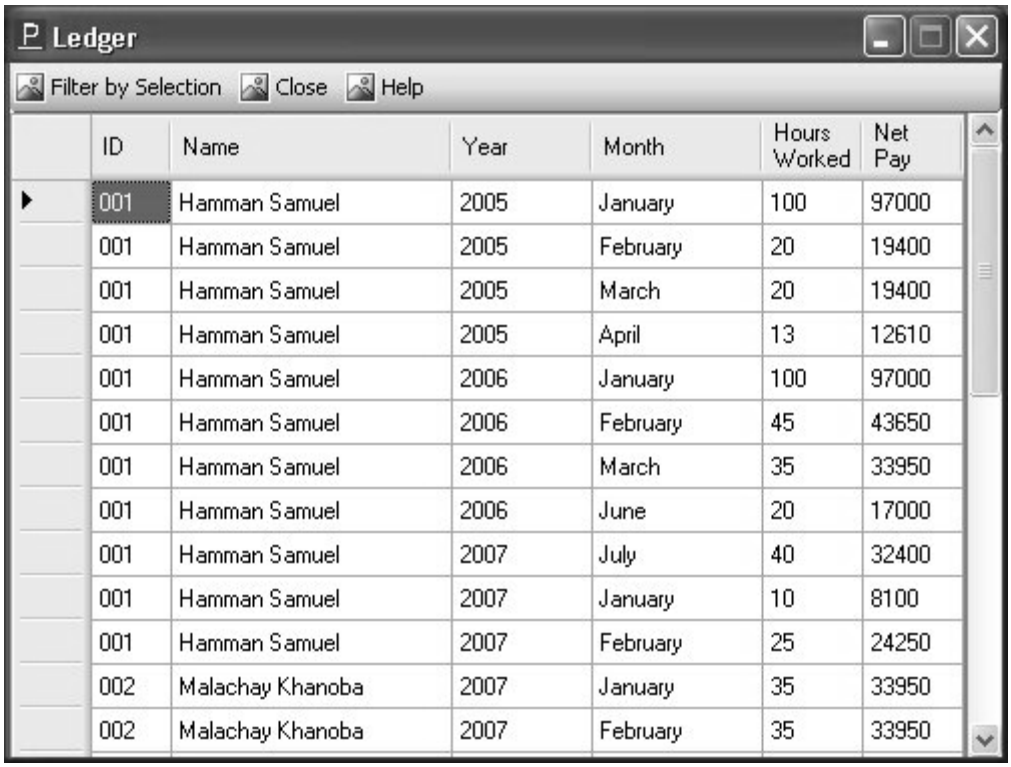
The Employee data displays the names and certain details about the employee.



This is the Payroll records which show all the information about the employee a selected employee.

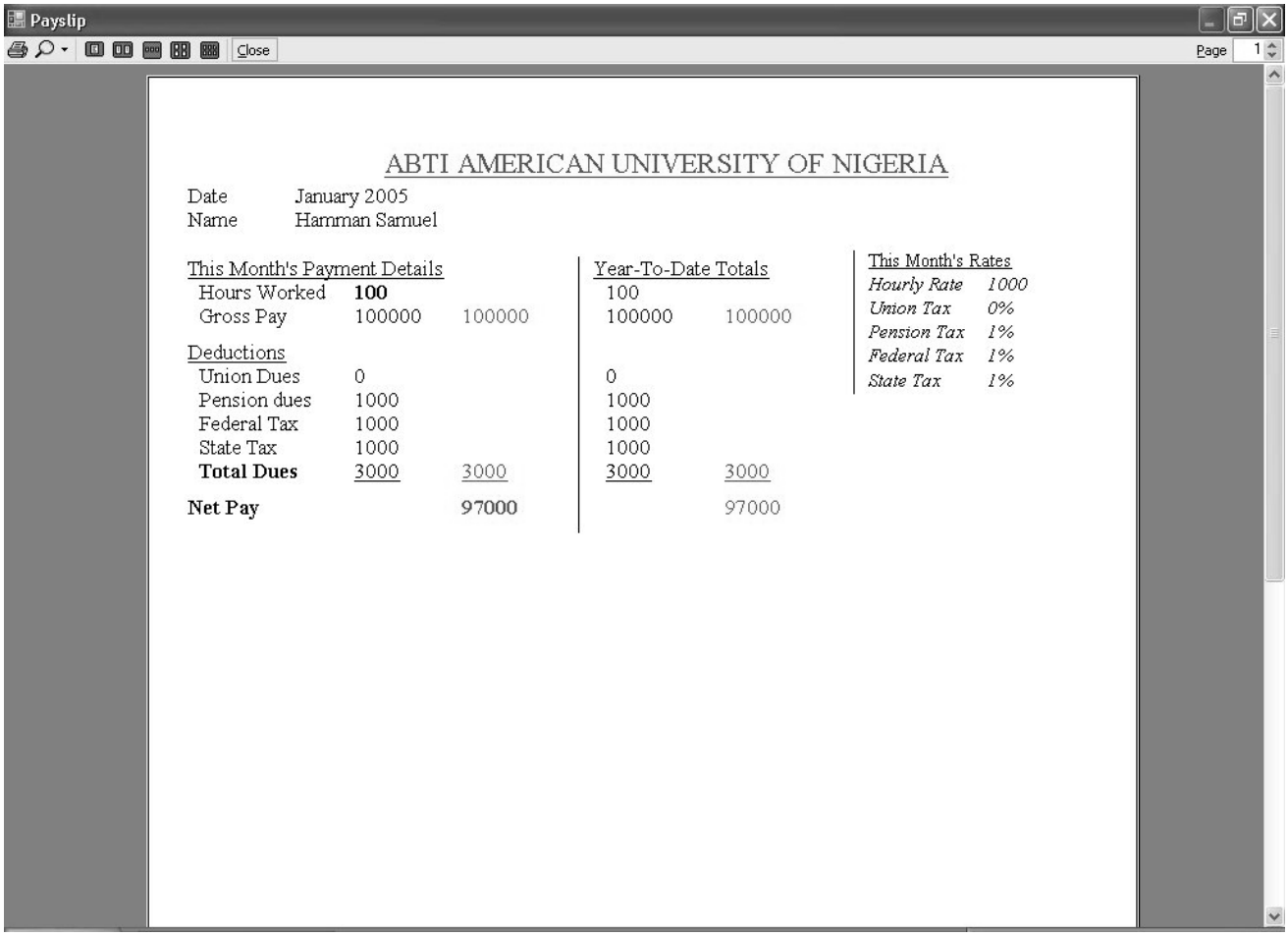


The Ledger screen shows a summary of all payrolls, and can be used to sort and filter the data to get a better view of data.

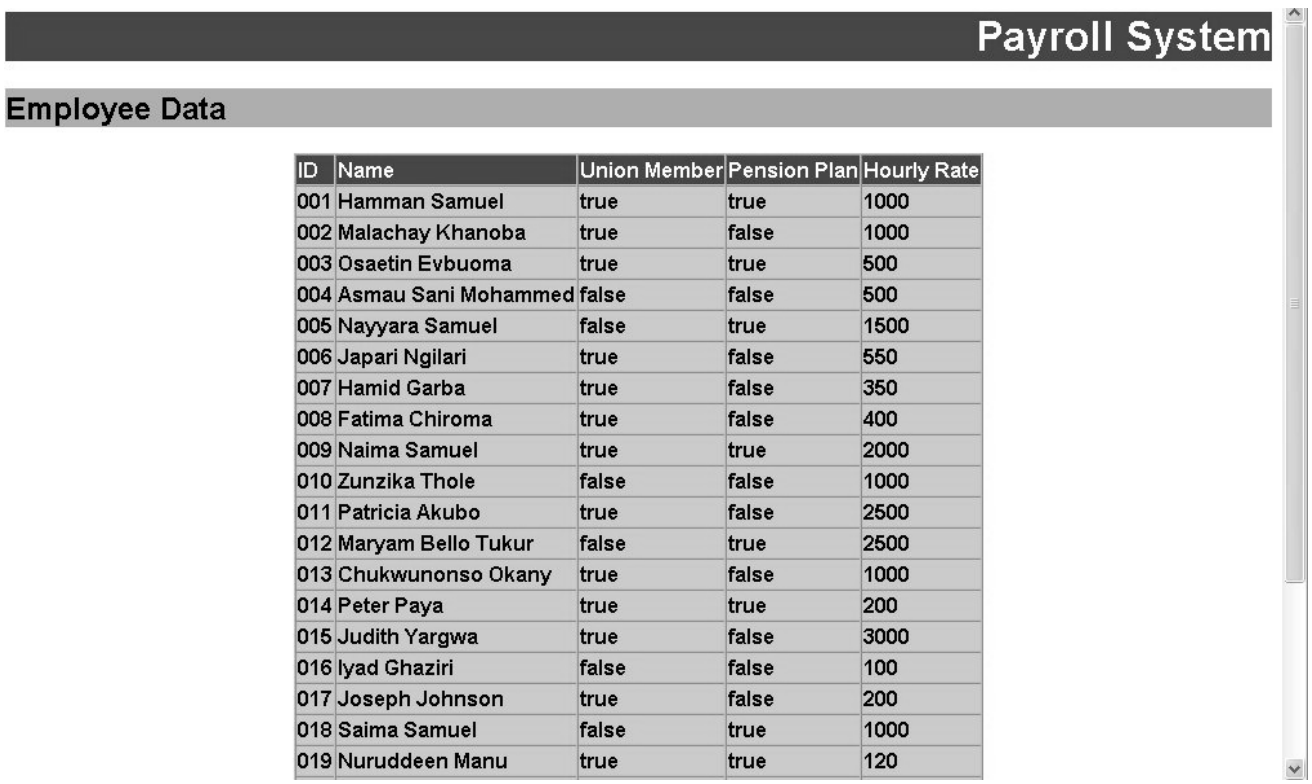


ID	Name	Year	Month	Hours Worked	Net Pay
001	Hamman Samuel	2005	January	100	97000
001	Hamman Samuel	2005	February	20	19400
001	Hamman Samuel	2005	March	20	19400
001	Hamman Samuel	2005	April	13	12610
001	Hamman Samuel	2006	January	100	97000
001	Hamman Samuel	2006	February	45	43650
001	Hamman Samuel	2006	March	35	33950
001	Hamman Samuel	2006	June	20	17000
001	Hamman Samuel	2007	July	40	32400
001	Hamman Samuel	2007	January	10	8100
001	Hamman Samuel	2007	February	25	24250
002	Malachay Khanoba	2007	January	35	33950
002	Malachay Khanoba	2007	February	35	33950

The payslip is printable and shows the current month's payment details as well as the year-to-date total.



There are various other web pages that are generated from the XML data directly.



ID	Name	Union Member	Pension Plan	Hourly Rate
001	Hamman Samuel	true	true	1000
002	Malachay Khanoba	true	false	1000
003	Osaetin Evbuoma	true	true	500
004	Asmau Sani Mohammed	false	false	500
005	Nayyara Samuel	false	true	1500
006	Japari Ngilari	true	false	550
007	Hamid Garba	true	false	350
008	Fatima Chiroma	true	false	400
009	Naima Samuel	true	true	2000
010	Zunzika Thole	false	false	1000
011	Patricia Akubo	true	false	2500
012	Maryam Bello Tukur	false	true	2500
013	Chukwunonso Okany	true	false	1000
014	Peter Paya	true	true	200
015	Judith Yargwa	true	false	3000
016	Iyad Ghaziri	false	false	100
017	Joseph Johnson	true	false	200
018	Saima Samuel	false	true	1000
019	Nuruddeen Manu	true	true	120

CHAPTER 4
RESULTS AND EVALUATION

4.1 Adequacy and Coverage

The payroll software can perform the following functions

- Store employee information
- Calculate gross and net pay, and determine tax deductions to be made
- Print payslips for each month, showing year-to-date totals
- Create and maintain a ledger containing all necessary records of employee payments
- Provide the user with adequate help by the user manual with the software

4.2 Efficiency and Effectiveness

- The payroll software calculates the total earnings of the employee and automatically updates the employee's earning to date.
- The employers can set different rates for employees.
- The software does its calculations in a very clear and concise manner. All calculations are guaranteed accuracy.

4.3 Productiveness

- The payroll software gives the employee the ability to keep track of their earnings by printing their payslip for each month.
- It gives the employers the ability to keep records of how much they pay out as salaries by creating a ledger that can be used to filter out results by employee, month, and year.

4.4 Elegance and User-friendliness

- The different tasks and functions are outlined in a very simple and clear manner for the users.
- The help file can be used by the users to know how to use the payroll software.
- The interface is very simple and not complicated to allow for easy usage.

4.5 Quality assurance

- The software was fully tested to ensure it is relatively error free and that it computes results correctly, including deductions such as taxes.
- The software provides a better, time saving and efficient way of keeping track of employees earnings by speeding up calculations, and reducing paperwork by keeping efficient electronic records.

4.6 Critical Evaluation

- The software overwrites employees information when re-entered. However, employee data is discarded when the employee is deleted from the software.

- The software also stores every information provided by the user, but does not store results of calculations. Instead, calculations are done “on the fly” when the user needs to see them on the ledger, the payroll records, or the payslip.
- Deductions such as federal tax, state tax and union tax are made according to the set rate provided by the employer, but additional benefits and bonuses are not included in calculations or storage of records.
- The payslip not only provides the employee with their earning, it also reflects their year-to-date recorded earning. However, there are no reports that explicitly summarize grand totals grouped by employees, months, or years. The ledger can be used to filter out these records.

CHAPTER 5
CONCLUSIONS AND FURTHER WORK

5.1 Summary

The main aim of this project was to put what we learnt in our Software engineering class into practice. The payroll system designated to our team allowed us to fully exercise the techniques of XP. The final deliverable was a simple payroll calculator and we were able to learn a new programming language, VB. In addition, we were able to apply the knowledge of OOP learnt in our Java classes to another language thereby giving us a better understanding of OOP.

5.2 Overview and Interpretation of Results Attained

We were able to attain our set objectives, and this helped us gain confidence in writing our own code and our own applications.

In addition, the use of serialization was an experimentation to cut down the time taken in designing the front-ends and back-ends of applications separately. With our OOP approach and serialization, we only concentrated on designing the objects/ classes, and then just serialized them on disk. So we did not spend any time on designing how to store data.

We also worked as a team, and gained some experience on how professional programmers work in the industry.

5.3 Recommendations on Future Improvement

There is always room for imporvement, and the software we created can also be imporved. This is especially because we had to create it within a limited time. With more time, the software can be improved to include security and different types of users. This would be the first step in making the software network-enabled, and eventually web-enabled. This was our original after-thought to programming the software, and we had chosen XML. In addition, the software can also be improved in terms of the calculations it can do, and more flexiblility in the rates used in calculations per employee.

REFERENCES

Horton, Ivor Java 2 SDK 1.4 Edition Wiley Publishing Inc., Indianapolis 2003

Millspaugh, Anita C. and Julia Case Bradley Programming in Visual Basic 6.0 Update Edition McGraw-Hill Irwin, Boston 2002.

Odinma, Augustine Software Engineering I SEN 261 Lecture Notes April/Spring 2007 ABTI-American University of Nigeria

Palmer, Daniel W. and Daniel H. Steinberg Extreme Software Engineering A hands-on approach Pearson – Prentice Hall, New Jersey 2004