ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

**School of Electrical Engineering and Computing**

**Program of Computer Science and Engineering**

**Semester Project**

I Revenue and Custom Automation

Henok Fekade

Gemechu Mulugeta

Natnael Alemu

Kaleab Alebachew

Fituma Amsalls

Submitted to:

Endale Aragu

Table of Contents

[Chapter One 4](#_Toc3762111)

[1.1. Introduction 4](#_Toc3762112)

[1.2. Background of the project 5](#_Toc3762113)

[1.3. Statement of the problem 5](#_Toc3762114)

[1.4. Purpose of the project 6](#_Toc3762115)

[1.5. Objective of the project 6](#_Toc3762116)

[1.5.1. General Objective: 6](#_Toc3762117)

[1.4.2. Specific Objective: 6](#_Toc3762118)

[1.6. Feasibility 6](#_Toc3762119)

[1.6.1. Economic Feasibility 6](#_Toc3762120)

[1.6.2. Technical Feasibility 7](#_Toc3762121)

[1.6.3. Operational Feasibility 7](#_Toc3762122)

[1.7. Scope and Limitations 7](#_Toc3762123)

[1.7.1. Scope 7](#_Toc3762124)

[1.7.2. Limitations 7](#_Toc3762125)

[1.8. Significance of the project 7](#_Toc3762126)

[1.9. Methodology 7](#_Toc3762127)

[1.9.1. Interviewing 7](#_Toc3762128)

[1.9.2. Practical Observation 8](#_Toc3762129)

[1.10. Development tools 8](#_Toc3762130)

[Chapter Two 9](#_Toc3762131)

[2. Description of existing system 9](#_Toc3762132)

[2.1. Major function of the existing system 11](#_Toc3762133)

[2.2. Drawback of the current system 13](#_Toc3762134)

[Chapter Three 14](#_Toc3762135)

[3. Proposed System 14](#_Toc3762136)

[3.1. Overview 14](#_Toc3762137)

[3.2. Functional Requirement 14](#_Toc3762138)

[3.3. Non-Functional Requirement 15](#_Toc3762139)

[3.4. System model 16](#_Toc3762140)

[3.4.1. Scenarios 16](#_Toc3762141)

[3.4.2. Use Case Model 19](#_Toc3762142)

[3.4.3. Object model 23](#_Toc3762143)

[3.4.3.1. Class diagram 23](#_Toc3762144)

[3.4.4. Dynamic model 25](#_Toc3762145)

[3.4.4.1. Activity diagram 25](#_Toc3762146)

[3.4.4.2. Sequence diagram 32](#_Toc3762147)

[3.4.4.3. State chart diagram 38](#_Toc3762148)

[Chapter Four 42](#_Toc3762149)

[4. System Design 42](#_Toc3762150)

[4.1. Overview 42](#_Toc3762151)

[4.1.1. Purpose of the System design 42](#_Toc3762152)

[4.1.2. Design goal 42](#_Toc3762153)

[4.2. Proposed system architecture 42](#_Toc3762154)

[4.2.1. System process 43](#_Toc3762155)

[4.2.2. Request Processor 44](#_Toc3762156)

[4.2.3. Subsystem decomposition 44](#_Toc3762157)

[4.2.4. Hardware/ software mapping 45](#_Toc3762158)

[4.2.5. Component diagram 46](#_Toc3762159)

[4.2.6. Database design 47](#_Toc3762160)

# Chapter One

## Introduction

Tax and revenue system is at the heart of governance. It is the way citizens pay tribute to the government. Today the body that implements this system is called **Ethiopia Revenue and Customs Authority**. It’s not a single body per se but is a hierarchical body. At the top there is the Federal then regional then city wide then finally woreda.

The current methodology is that a person willing to open his own business first gets a license by registering as a merchant. Then he informs his local revenue center (local meaning the office responsible for collecting revenue of the area where he has opened up shop) that he has begun work. Then they open up a record and inform him the time period in which he must pay his taxes and the time period when he must come and verify his payments.

The problem with this methodology is that first of all its manual implementation makes it stressful in the time period the merchants come to verify their claim (and as we all know stress results in unintentional faults) and mismanagement of files and records happen causing problems for both the merchant and the employees working. Second of all analysis of the data is very tedious and unmanageable because of implementation of files only physically in a cabinet. And accessing the files only a few times degrades the file and its life time decrease with every use case.

This is the project aimed at this very issue. It is a way to automate the working process by implementing files as database in a computational machine. And implementing their working process as a software so that every action every stake holder takes will be computer aided resulting in a robust and error prone working environment.

The documentation to follow explains how our team took about the problem. It’s divided into four chapters each chapter growing one step forward to the solution.

* Chapter 1: Includes an introduction to the system and the working environment of the system and the team.
* Chapter 2: explains the current implemented system in detail. It determines the major functions and features of the system.
* Chapter 3: based on the results of chapter 2 models the system into a set of scenarios and determines the different use cases, the internal relations of the system and functions.
* Finally Chapter 4: describes how the team realized and deployed the system. It includes database design, hardware/software mapping and component diagram.

The models and diagrams used comply with UML representation and understanding of this models and design might require a background knowledge of Object oriented design, database design and software engineering.

## Background of the project

The tax and revenue system has existed since the beginning of government. It is the basis for the foundation of our modern day way of life. Tax and revenue gives government the financial power that enables it to guard our boundaries, secure our rights, and in every sense be someone each of us can rely on.

This system that enables government be governor is not as simple as it used to be; There are now, vast number of classification (it’s no longer as easy as saying “farmers must pay this much and merchants must pay this much.”), It’s no longer physical transaction it’s all abstract and complex.

The government treats this division separately and calls it “**The bureau of tax and revenue”.** It has a down tree hierarchy at goes from federal to regional down to woreda/ Kebele. And as such handle different classes of the population: Federal handles those businesses that have revenue above 10million, Regional handles those above 1million and below 10million and the rest is handled by woreda.

Since Federal and regional revenues consist most percent of the total revenue collected by government, a computer based, web interfaced, highly modern system is already in place and thus have a very efficient and reliable methodology in place. When we come to woreda level there is no such system, there is no methodology only dedicated employees trying their best to do their job (i.e. trying different approach to organize their paper-work (database), trying different approaches to interact with their customers…)

Even though most of the revenue collected tend to be in regional and federal level, most of the business’s (and thus most of the population) are in woreda level. Accordingly making their job a lot harder: they interact much more, have much more data, and are the most useful level for analysis and generating information, knowledge, and alike.

The manual system currently in play is growing out of date and becoming unable to satisfy both the citizen and the officials. Too much information loss, too much bureaucracy, and most of all so much unorganized data causing difficulty for processing and analysis.

This is where we come in. This project is directly aimed at this problem. This project is intended to upgrade the manual system by enabling the workers to include the aid of computers. Specifically we try to develop an application that automates their day to day manual work so that every information is under one roof (i.e. in a database), there is no longer misinformation (once the information is inserted in the system its visible to every one thus reducing misinformation), and most of all it will serve as the basic guideline enabling a standard and smooth interaction between the citizens and the officials.

## Statement of the problem

As wide and vast reaching this system might seem most of our society is involved in very low revenue rate that the system does not include. Like barber shops, breakfast houses, small shops, taxi, electronics repair shop … and so on. And the worst part is each of the examples listed above are distinct classes in the system and have different methodology and require different and special treatment; making the manual process of handling this troublesome and time consuming.

One, the data storage is only in hard copy (i.e. paper work) format making retrieving and updating a client’s data a time consuming nightmare, also causing data loss due to both degradation and human error.

Two, any conclusion or report to be made must first scan the entire archive manually, and it may not only even be once. If a perspective shift occurs along the way then the manual scan must be repeated. And depending on the size of the archive this takes days, weeks, or even months for a simple analysis.

Third, it’s becoming unmanageable, in a sense that the officials are unable to determine who has paid and who has not or who has completed the payment and who is still hanging. And from the citizen’s point of view its becoming difficult to argue with the officials because of difficulty of data retrieval and data consistency thus are being subject to extra payment or extra bureaucracy because of mislead information and lack of standard standing ground.

## Purpose of the project

* Creating an efficient and most of all effective data management system.
* Creating an efficient and more reliable data analysis system.
* Structuring and automating the current working culture.

## Objective of the project

### General Objective:

The general objective of this project is to design and develop an application software that understands the complex tax and revenue payment calculation system so that it can reduce the overload of organizing, analyzing, and storing of the data produced by the many citizens obliging to their constitutional obligations.

### Specific Objective:

1. Calculate the amount each individual must pay according to the payment system.
2. Store data in an efficient and modern way(i.e. in a database)
3. generate reports and present analyses based on the required criteria

## Feasibility

In determining whether our proposed system can be realized. We analyzed 3 different types of feasibility tests.

### Economic Feasibility

This is to determine the benefits that are expected from a proposed system and compare them with costs. And make sure that the costs are payable and in the end make sure it was all worth it.

For the completion of this project no or very low cost (cost of transport) is required.

And in the implementation stage no new hardware is required. Our application only needs a working, connected desktop; which our target clients already have.

So YES it is economically feasible.

### Technical Feasibility

Technical in a sense, evaluating if the current technology has a potential to support our proposed system.

The answer is a big YES as mentioned above no new technology or software is required and thus it is technically feasible.

### Operational Feasibility

By operational we mean how well a proposed system solves the problems, and how it satisfies the needs of the clients practically.

In every way possible it is made to be a user friendly, straight forward, user interactive application that is accessible by any one.

## Scope and Limitations

### Scope

* The system only considers the woreda level working process
* Developed only from employees point of view and thus not much concern given to the clients point of view

### Limitations

* Is a new implementation and requires user’s adaptation
* It’s an automation process thus may cause a few peoples out of job

## Significance of the project

Significances of this System are:

####  minimizing the time wasted to find different services (for the tax payer).

 providing information for users about the variety of payment they are paying.

 minimizing the energy wasted to analyze data (for the officials).

## Methodology

We plan to use Waterfall modeling system because

1. We are working on a well-defined and known working system and we simply wish to automate it.
2. Second as of our time constraint, it offers the least interaction with our stake holders.
3. The modeling system offers the best means for tracing our change and documenting it

### Interviewing

As been described this is a well-known and defined system and thus for us to understand it there is no better way than asking from the working employees.

### Practical Observation

We tried to model their daily routine by observing most of their daily work, and by observing the different scenario’s they conducted on our behalf.

## Development tools

In ideal case this system is best suited for web-app (i.e. using PHP, html, Xamp, css…). Thus our development tool of choice will be just that.

Our database server will be managed using MySQL program

The design and architecture of our system will be aided by Enterprise Architect & Visual Paradigm

More or less we will be using a computer with platform Windows for realizing this system.

# Chapter Two

## Description of existing system

A pointer here is that as a general tax and revenue system there exist a fully automated, very modern computer based system; but its application is only in federal and regional level, and those that are treated at regional and federal level are those earning more than 1 million birr revenue per year and discludes those that do not meet this threshold. Thus the remaining citizens are treated manually and records exist in an old fashion file system in only paper and shelf.

Let’s start with a simple but very abstract model. We can describe the existing system by two main functionalities, namely: keeping track of local merchants AND collection and verification of tax and subsequent payments. Let’s refine this functionalities one by one.

When we say ‘keep track of merchant’, means the system keeps records of the merchant’s activities and updates them as time goes by and as per the activity of the merchant.

A merchant, as understood by the system, is a person who has registered legally for a specific type of work and has acquired a tin number. Every such merchant, after getting their tin-number, must report to their local revenue center and open a record. The record consists of their personal detail (i.e. Full Name, tin-number, place of residence,) and work detail (i.e. type of work, expected annual return, the place where the work gets conducted, the date when the work started,). Then after, the merchant is required to come and update his status at least once a year within a specified time period (the system posts the time frame in which the specified type of merchant’s must come and update their status).

Each year the merchant comes to the local revenue center with a bank receipt that verifies that he/she has fully paid their annual tax and subsequent payments. Some times what the merchant thought was his full payment and therefore paid, turns out to be less than what is expected of him. In such cases the merchant has one of two choices, either he/she go finish the payment and come back with the new receipt that verifies that he/she has fully paid their annual tax or they can request to finish the remaining amount till next year.

The center also performs routine and surprise audits in its region of authority. By audit we mean that an auditor or auditors go around and checks if what the merchant is claiming to do (as what the record says) matches with what is truly going on. This check extends to verifying their books such that the auditor can verify, what they claim to be their annual return is their true and practical return. This audits often change the status of many merchants. The year that was last audited is saved as a record. And whenever this audits are conducted the auditor opens the book (the journal of the merchant) starting from the last audited year.

When a merchant-for whatever reason-wants to stop working, he/she is required to come and close their record. To do so the system checks to see that the merchant has fully finished his/her payment-which is calculated till the present closing day-and that he has no remaining balance (i.e. has no debit nor any remaining balance that the system owes him). If all checks out his/her record is now considered dead and the system no longer expects any updates from this record.

When we say ‘collection and verification of tax and subsequent payments’, we imply the underlying calculation of merchant’s required payment amount by taking as input his/her data and a calculation sheet which is supplied from the regional office.

The currently existing system categorizes its clients in to 2 main groups, namely: those that book their every transaction (and thus have cash register) and those that have predetermined annual revenue. Let’s call them books and Kurt’s respectively for simplicity of discussion.

The way the Kurt system work is once every 4 or 5 years an official circulates and observes the size of the registered business and sets their total annual revenue. Doesn’t matter if that business blooms or dies after that, till the next circulation they must pay according to that fixed value, there are mainly 3 type of payment they must pay: 1. Tax

2. Tot or VAT

3. Sales

And 2 special case payments: Penalty and Special. And each has different calculation for different level of revenue (more on each type of payment later)

Handling books is a simple and straight forward process. They prepare there quarter annual, semiannual and annual income statement and balance sheet- which are organized form of every single action they have made within the time period and include the total payment they ought to pay. Thus the center’s job will be to verify the validity of the presented calculation and record it on the extension of the regional’s system (of course they also keep a hard copy record).

Again the population can be divided into 3 different classes. 1. Class A

2. Class B

3. Class C . this classification is based on amount of annual revenue. And mostly Revenue and Custom’s Authority centers use this classification for management purpose. In other words the time period within the year, that merchants must pay is divided according to this classification.

Let’s say a few things about the type of payments:

1. Tax: this is one of the major type of payments that everyone is expected to pay. It’s usually calculated as some percent of the annual return. Since our country uses progressive tax system, those earning more are obliged to pay a higher percentage. Since this is not a fixed type of payment (i.e. may vary from year to year according to the countries state) the 'who pays how much' or calculation sheet transits from federal to regional to city and finally to woreda, which they implement on the merchant.

2. TOT/VAT: this is also one of the major type of payments that everyone should pay. But here one pays either one or the other (either TOT or VAT) never both. The 'who pays TOT and who pays VAT ' question depends on type of work. Meaning fields of work that are considered services instead of product offering are obliged to pay VAT (like Electronics warehouses, Barbershops, Pool houses,...) and those earning more than 100,000$ annually no matter type of work must also pay VAT. The rest pay TOT. Both are of fixed type and value in a sense that they don't vary with time and also don't need any 'calculation sheet' TOT is always 2% and VAT is always 15%.

3. Sales: this is also one of the major types and this is dedicated for clean-up purposes. Or this is the price the merchant pays for a clean city and consequently a clean workplace. This is also not of fixed type and thus the calculation sheet is expected from the regional government to be implemented.

4. Penalty: as its name implies is a deduction made on those who miss the deadline to update their status. And thus only late comers pay this.

5. Special: this is a special type of deduction on those businesses that have employees. If the business uses a book (i.e. keeps a journal of its activities) then the name special need not be applied. Because this deduction methodology corresponds to the tax imposed on the employees, the journal will include the salary paid to the employees and their resulting payments (a pointer here is this also requires a calculation sheet which as usual the region provides). What makes it special is its mostly imposed on those that don't use book (like shops, taxis,..) and thus is an estimate that says " taxis pay '...' much for their associates and thus every taxi of '....' size must pay an additional '.....' much" surprisingly this also needs a calculation sheet that is provided by the region and therefore not the woreda's job to make an estimate.

If this much is said for the type of payments. Let’s say a little more on the practical work culture. With no data one may expect that all type of works are equally treated but the reality is some fields dominate. Like for example here in Adama city transportations (including minibus's and bajaj's) dominate of all else. And from transportations bajaj's dominate and the culture is to have bajaj's record closer on hand than the rest and are referred to as '01' and the rest of the transportations like minibuses, buses, Isuzu’s, water trucks,... every else is referred to as '03'. And by the above reasoning bajaj's are given physically close and front archive numbers (which is a serial number generated for every record for storage and retrieval purposes).

One other culture is those used for all type of transportation. Mostly all transportations are of type Kurt. And accordingly there exist a calculation sheet for all type and level of transportations. Starting from Bajaj, minibus, to water truck, agricultural tractor, construction tractor, and etc.… Making the management of transportations predetermined, plus being the most popular field of work makes them a separate class.

One last concept that cannot be skipped is the concept of ‘Xamraa’, which is a name given for a merchant that has deployed on two types of works. Since such merchants have one record and one tin-number for both their works they are handled separately and are given the name 'Xamraa'

### Major function of the existing system

1. Registration of merchants: currently the process is

1st the merchant is prerequisite to have a tin-number (which is given at higher levels of government.

2nd the merchant comes with hard copy proofs of his tin number, working place rent agreement (or any form of proof of workplace)

3rd a record (in a form of a file) is opened in his name and he is given an archive number to remember when he comes next

1. Debit&Credit Management: Manage a debit and credit system: means allow a way for merchants to pay at a later time, as discussed above, which is debit. And some pay more this year- may be because business boomed- and store the remaining for next year. Which is credit.
2. Time and notification management
3. Organization of information to best fit the time and need: means for example at verification types if it is the time period for Class A merchants, then Class A merchant’s information is closer at hand and all the rest is put under the desk. And according to the time penalties may apply thus a management system is needed.
4. Notification means addressing information to the merchant. It may be the payment period, or any changes made to the working process, or any change that directly concerns the merchant must be addressed effectively and in the shortest amount of time.
5. Verification of payments: means two things
6. Make sure what the merchant paid matches the payment obliged.
7. Make sure the payment is legit (authenticate validity of receipt) or that the payment has truly been addressed to them.
8. Audits and inspections: which includes the routine and surprise checkups done on validity of information. Which is mostly for those who use books. And the analysis, processing and storage of such audits is done by an extension of the regions online system. Our system only records that such an audit has occurred and remembers the date it was conducted.

.

1. Payment calculation: means the actual implementation of the different calculation sheets. This is the heart of the system and the most important functionality. First determines which merchant pays which payment. And then determines which case applies to him by examining the calculation sheet. Finally executes those specifications to determine the total payment addressed by the merchant.
2. Keep record: means keeping proof of everything that happened. Whenever a merchant pays, or changes his status Or whenever a federal or regional change happens on the working process this changes must be recorded properly.

### Drawback of the current system

1. Data retrieval, analysis, or any type of management is physical record dependent thus , records \*wears out

\*get misplaced

\*or even get lost

1. Manual system of payment verification opening way for frode and unreliable system.
2. Lack of accuracy and precision (sometimes to the point of unreliability) in payment calculation.
3. No defined and working structure, mostly work culture dependent. Causing communication problems
4. No feedback loop nor any sort of knowledge retrieval. Leading to a degenerating system.

# Chapter Three

## Proposed System

### Overview

The proposed system aims at automating the current, woreda level, revenue system. It records and organizes data in a well-designed database. It removes from the old system all the technical problems, all the unnecessary manual verification and allows data access and analysis on a level that was not possible before. Furthermore it breaks down the complex payment system to a level where a customer with no background can easily understand and therefore easily communicate with the employees.

## Functional Requirement

* **Register new merchants**

**The System will register merchants with in the woreda region which have acquired tin number as id from the federal bureau of revenue.**

* **Verification of payment**

**Verify that each merchant has paid what he supposed to pay.**

* **Calculate how much a merchant should pay and why**

**For every merchant with a unique tin-number, breaks down the calculation and presents in an easy format understandable by any one.**

* **Provide a managerial report**

**That is, updates the manager on who has paid and who has not, how much has been collected and how much should have been collected, and breaks down the collected money according to payment system (i.e. money from vat, money from tax, money from tot...).**

* **Open and close records**

**Merchant can stop work (i.e. close his record) or can restart his work (i.e. open his record)**

* **Updating calculation sheet**

**Supports any high level changes made to how they should proceed.**

* **Change Kurt’s status**

**Changes Kurt’s expected annual return.**

* **Exceptional merchants management**

**Supports Xamraa and those with credit and debit**

* **Employee Management**

**Includes hiring new employees and firing existing**

* **Audit**

**Recognize that an audit has been made**

* **Notification**

**Addressing change to the person of interest and making sure information flow is maximum.**

* **Record**

**Storing of dynamic change made to the working process or to the merchant.**

## ****Non-Functional Requirement****

**Usability**

The system should be easy to learn and understandable for all the users

**Reliability**

The system should be highly reliable. Since the user of the system is a government office, that deals with money the system should be very accurate to the maximum available precision. And must not at all have any glitches or confusions or bugs of any sort. Because a little mistake results in a very large financial crises for many parties.

**Performance**

The system should respond within a short period of time. It depends on the performance of the hardware environment. As a governmental system it should perform better to maximize its productivity.

**Compatibility**

The system should have to be compatible; the application is developed having windows operating system environment in mind, since that is the operating of almost all computers in our country.

**Accessibility**

The system should be accessible at any time. Our system has no dependability’s except for the operating hardware and thus is as available as the hardware is.

**Maintainability**

The system should be maintainable. As it is

**Error Handling**

Each error that may occurs will be handled accordingly in order to reduce the amount of failure. Since users of the system are human, error is inevitable, and thus each and every input will be handled having that in mind.

**Provides Security**

**The system provides a hierarchical authentication system, allowing only appropriate users to use the appropriate functionalities.**

## System model

### Scenarios

**Scenario: 1**

**Name of use case:** Register merchant

**Participating instance actor:** employee and merchant

**Entry condition:** Employee has already logged into the system

**Flow of events:**

1. Merchant comes and requests registry
2. Employee verifies that he is qualified
3. Employee Selects “Register” button
4. System displays the registration form
5. Employee fills in the form inquiring merchant
6. and clicks “Submit” button

**Alternate conditions:**

* Employee fills the form incorrectly, generating an error message
* Merchant is not valid candidate for register and thus employee doesn’t invoke system
* The merchant is already in the system, also generating an error message

**Exit condition:** The system saves the entered data into database

**Scenario: 2**

**Name of use case:** Verification of payment

**Participating instance actor:** Employee and merchant

**Entry condition:** None

**Flow of events:**

1. Merchant arrives with receipt of payment
2. Employee verifies validity of receipt
3. Inputs merchants unique tin-number in system
4. Invokes **calculate payment**
5. Checks if paid amount equals calculated amount
6. If equal Updates Merchants status by clicking **paid** button
7. If not equal invokes **manage debit and credit** by clicking **debit/credit** button

**Alternate conditions:**

* Merchant comes with a fake receipt resulting in the merchant being escorted out

**Exit condition:** The system updates the change to the database

**Scenario:** 3

**Name of use case:** Calculate Payment

**Participating instance actor:** Employee

**Entry condition:**

* Employee already inputted merchant’s tin-number

**Flow of events:**

1. Employee clicks Calculate Payment button
2. Generates a table of merchant’s obligated payments with their amount and a total sum at the end.

**Alternate Conditions:** None

**Exit condition:** The system updates the change to the database

**Scenario: 4**

**Name of use case:** Generate report

**Participating instance actor:** Manager

**Entry condition:** None

**Flow of events:**

1. Manager clicks the “report” button
2. Select type of report
3. Select the scope of the report
4. Enjoy

**Alternate Conditions:** None

**Exit condition:** displays report

**Scenario: 5**

**Name of use case:** Open/Close record

**Participating instance actor:** Employee and merchant

**Entry condition:** Employee has already logged into the system

**Flow of events:**

|  |  |
| --- | --- |
| 1.Merchant requests to close record | 1.Merchant requests to open record |
| 2.Employee enters merchant’s tin-number | 2.Employee enters merchant’s tin-number |
| 3.Clicks open/close button | 3.Clicks open/close button |
| 4. If merchant has a valid status, prompts to close | 4.Prompts “Successfully opened” |
| 5.Employee clicks close button |  |
| 6.Prompts “successfully closed” |  |

**Alternate Conditions:**

* If merchant doesn’t have a valid status, displays reason

**Exit condition:** The system updates the database

**Scenario: 6**

**Name of use case:** Update calculation sheet

**Participating instance actor:** Manager

**Entry condition:** Manager has already logged into the system

**Flow of events:**

1. Manager clicks Update button
2. Chooses which calculation sheet to update
3. Fills in the new calculation sheet
4. Clicks **save** button

**Alternate Conditions:**

* Updated fields defy application logic then returns error message

**Exit condition:** The system saves the entered data into database

**Scenario: 7**

**Name of use case:** Change Kurt’s status

**Participating instance actor:** Official and merchant

**Entry condition:** Official has already logged into the system

**Flow of events:**

1. Official requests merchant’s tin-number and inputs it.
2. display’s merchant’s current expected annual return
3. Enters new annual return
4. Clicks **Save** button

**Alternate Conditions:** None

**Exit condition:** The system saves the entered data into database

**Scenario: 8**

**Name of use case:** Audit

**Participating instance actor:** Employee

**Entry condition:** Employee has already logged into the system

**Flow of events:**

1. Employee enters merchants tin-number
2. Clicks Audit button
3. System saves current date as last audit

**Alternate Conditions:** None

**Exit condition:** The system updates database

**Scenario: 9**

**Name of use case:** Employee Management

**Participating instance actor:** Manager

**Entry condition:** Manager has already logged into the system

**Flow of events:**

1. Manager clicks employees button

|  |  |
| --- | --- |
| 2.Clicks the hire button | 2.Clicks the fire button |
| 3.Fills employees information on the form displayed | 3.enters employee id on the form shown |
| 4.Clicks save button | 4.Clicks fire button |
| 5.System generates username and password for new employee and displays it | 5.prompts fired |

**Alternate Conditions:** None

**Exit condition:** The system updates database

### Use Case Model

#### Actor identification

* **Manager**
* Login
* Update calculation sheet
* Employee Management
* Generate report
* Logout
* **Employee**
* Login
* Register merchant
* Verify payment
* Open/Close records
* Manage credit and debit
* Calculate Payment
* Audit
* Logout
* **Officials**
* Change Kurt’s status
* **Merchants**
* View status
* Get notification

#### Use case identification

* Login
* Register merchant
* Verify payment
* Update calculation sheet
* Generate report
* Open/Close records
* Manage credit and debit
* Change Kurt’s status
* Employee management
* Audit
* Logout
* Calculate payment
* Get notification
* View status

#### Use case diagram

Figure 1 use case diagram

#### Use case description

|  |  |
| --- | --- |
| Use case name | Login |
| Use case description | To authenticate users |
| Actor | Manager, Employee, Official |
| Pre-condition | User must have username and password |
| Post-condition | Login to respective page |
| Main flow | 1. Actor open the system 2. The system display login form. 3. User fills correct username and password. 4. Click login button. 5. The system redirect to the index page |
| Exceptional flow | Incorrect user/password will generate an error message |
| Frequency of use | Often |

Table 1 use case description for login

|  |  |
| --- | --- |
| Use case name | Register merchant |
| Use case description | To register new merchant into the database |
| Actor | Employee |
| Pre-condition | Employee must have logged into system |
| Post-condition | System successfully registers and displays merchant’s information |
| Main flow | 1. Employee Selects “Register” button 2. System displays the registration form 3. Fills in the form and clicks “Submit” button |
| Exceptional flow | Employee fills the form incorrectly, generating an error message  The merchant is already in the system, also generating an error message |
| Frequency of use | Moderate |

Table 2 use case description for Register merchant

|  |  |
| --- | --- |
| Use case name | Verify Payment |
| Use case description | To make sure proper payment has been made |
| Actor | Employee |
| Pre-condition | Employee must have logged into the system |
| Post-condition | System successfully updates the database |
| Main flow | 1. Employee Inputs merchants unique tin-number in system 2. clicks calculate payment button 3. Checks if paid amount equals calculated amount 4. If equal Updates Merchants status by clicking paid button 5. If not equal, clicks debit/credit button |
| Exceptional flow | None |
| Frequency of use | Moderate |

Table 3 use case description for verify payment

|  |  |
| --- | --- |
| Use case name | Calculate Payment |
| Use case description | To generate the payments expected of the respective merchant |
| Actor | Employee |
| Pre-condition | Employee must be logged in and has already entered merchant’s tin-number |
| Post-condition | Generates and displays a tabulated form of payments required by current merchant |
| Main flow | 1. Employee clicks Calculate Payment button 2. Generates a table of merchant’s obligated payments with their respective amount and with a total sum at the end |
| Exceptional flow | None |
| Frequency of use | moderate |

Table 4 use case description for Calculate Payment

|  |  |
| --- | --- |
| Use case name | Manage employee |
| Use case description | Displays a merchants bill information |
| Actor | Manager, employee, merchant |
| Pre-condition | None |
| Post-condition | None |
| Main flow | 1. Click the “display” button 2. Fill in merchant’s unique tin-number 3. Enjoy |
| Exceptional flow | 1.Incorrecte tin-no will generate an error message |
| Frequency of use | Often |

Table 5 use case description for Viewing merchant’s payment information

|  |  |
| --- | --- |
| Use case name | Open/Close record |
| Use case description | Merchant updates his status to either “Starting to work again” or “Stopped working” |
| Actor | Employee |
| Pre-condition | Employee has already logged into the system |
| Post-condition | Updates the database |
| Main flow | |  |  | | --- | --- | | 1.Employee enters merchant’s tin-number | 1.Employee enters merchant’s  tin-number | | 2.Clicks open/close button | 2.Clicks open/close button | | 3. If merchant has a valid status, prompts to close | 3.Prompts “Successfully opened” | | 4.Employee clicks close button |  | | 5.Prompts “successfully closed” |  | |
| Exception flow | If merchant doesn’t have a valid status, displays reason |
| Frequency of use | Rare |

Table 6 use case description for Open/Close record

|  |  |
| --- | --- |
| Use case name | Generate report |
| Use case description | Provide analyzed information |
| Actor | Manager |
| Pre-condition | Manager must be logged on |
| Post-condition | Displays desired analyzed information in table format |
| Main flow | 1. Selects “report” button 2. Select type of report 3. Select the scope of the report 4. Enjoy |
| Exceptional flow | None |
| Frequency of use | Often |

Table 7 use case description for Generate report

|  |  |
| --- | --- |
| Use case name | Logout |
| Use case description | To leave page |
| Actor | Manager, Employee, Official |
| Pre-condition | User must have username and password |
| Post-condition | User leaves the page |
| Main flow | 1. User clicks logout link 2. The actor leaves the page |
| Exceptional flow | None |
| Frequency of use | Often |

Table 8 use case description for Logout

|  |  |
| --- | --- |
| Use case name | Update calculation sheet |
| Use case description | Updates the current way of calculating payments |
| Actor | Manager |
| Pre-condition | Manager must be logged on |
| Post-condition | Updates the database |
| Main flow | 1. Manager clicks Update button 2. Chooses which calculation sheet to update 3. Fills in the new calculation sheet 4. Clicks savebutton |
| Exceptional flow | Updated fields defy application logic then returns error message |
| Frequency of use | Rare |

Table 9 use case description for Update calculation sheet

|  |  |
| --- | --- |
| Use case name | Change Kurt’s status |
| Use case description | Updates the status of the merchant |
| Actor | Official |
| Pre-condition | Official must be logged on |
| Post-condition | Updates the database |
| Main flow | 1. Official inputs merchant’s tin-number. 2. display’s merchant’s current expected annual return 3. Enters new annual return 4. Clicks Savebutton |
| Exceptional flow | None |
| Frequency of use | Rare |

Table 10 use case description for Change Kurt’s status

|  |  |
| --- | --- |
| Use case name | Audit |
| Use case description | Updates the status of merchant |
| Actor | Employee |
| Pre-condition | Employee must be logged on |
| Post-condition | Updates the database |
| Main flow | 1. Employee enters merchants tin-number 2. Clicks Audit button 3. System saves current date as last audit |
| Exceptional flow | None |
| Frequency of use | Rare |

Table 11 use case description for Audit

### Object model

### Class diagram

We cannot fully describe the class diagram with just one diagram. So we have decomposed the diagram. The first figure will show the top view and will only contain the relationship of the classes and the figures that follow will contain portions of the top view but this time containing details of each class.

Figure 2 class diagram (top view)

## Dynamic model

### Activity diagram

1. **Login**



Figure 3 Activity diagram for Login

1. **Register Merchant**



Figure 4 Activity diagram for Register Merchant

1. **Verify payment**



Figure 5 Activity diagram for Verify payment

1. **Calculate Payment**



Figure 6 Activity diagram for calculate payment

1. **Open/Close record**



Figure 7 Activity diagram for Open/Close record

1. **Update Calculation sheet**



Figure 8 Activity diagram for Update calculation sheet

1. G**enerate Report**



Figure 9 Activity diagram for generate report

1. **Audit**



Figure 10 Activity diagram for Audit

1. **Change Kurt’s status**



Figure 11 Activity diagram for Change Kurt’s status

### Sequence diagram

1. **Register merchant**

****

Figure 12 Sequence diagram for Register merchant

1. **Audit**

****

Figure 13 Sequence diagram for audit

1. **Verify Payment**

****

Figure 14 Sequence diagram for Verify Payment

**3. Generate report**

****

Figure 15 Sequence diagram for generate report

1. **Calculate Payment**

**** Figure 16 Sequence diagram for calculate payment

1. **Update Calculation sheet**

****Figure 17 Sequence diagram for update calculation sheet

1. **Change Kurt’s status**

****

Figure 18 Sequence diagram for change Kurt’s status

1. **Open/Close record**

****

Figure 19 Sequence diagram for open/close record

1. **Login**

****

Figure 20 Sequence diagram for login

1. **Logout**

****

Figure 21 Sequence diagram for logout

### State chart diagram

1. **Login**

****

Figure 21 State diagram for verify payment

1. **Register Merchant**

Figure 21 State diagram for verify payment

1. **Open/Close record**

Figure 21 State diagram for verify payment

1. **Update sheet**

Figure 21 State diagram for verify payment

1. **Calculate Payment**

Figure 21 State diagram for verify payment

1. **Verify Payment**Figure 21 State diagram for verify payment

# Chapter Four

## System Design

## Overview

This is the system design of Revenue and Customs Authority. So far we have been in the problem domain. System design is the first part to get into the solution domain in a software development.

System design part of the system is the transformation of the analysis model into a system design model. This chapter focuses on transforming the analysis model into the design model that takes into account the non-functional requirements and constraints described in the problem statement and requirement analysis sections discussed earlier.

This document describes the design issues of the overall system, such as design goal, subsystem decomposition, hardware/software mapping, and persistent data management. It provides the complete architectural overview of the proposed system. It is intended to capture and express the significant architectural decisions, which have been made, on the system

### Purpose of the System design

The aim of this design is to overcome the malfunction of the old system by using only available resources as a realization media. Next we explicitly specify the system required goals we followed to best optimize the realization.

### Design goal

The design goals are derived from the nonfunctional requirements. Design goals guide the decisions to be made by the developers when trade-offs are needed. And thus, as described in the previous chapter, the design goals are

1. Reliability: Its calculation should be accurate ( without round off to the last cent)
2. Simple Interface: As an automation, unless our users fill at home, the users will not judge it by its functionality

## Proposed system architecture

As a web application, we have decided to use a three-tier architecture, which has three layers. These three layers are the Application or Presentation layer, the business layer and the data access layer. Application or presentation layer is the form, which provides the user interface to the end user. In our case the client side processing or the pages displayed. The business layer is the class, which implements all the functionality of the system, which works as a mediator to transfer data from application layer or presentation layer to data layer, which is implemented as the server side services and in a way form the API of the system. The third tire is the data access layer which is also a class to get or set data to the database queries back and forth. This layer only interacts with the database. The database queries or stored procedures will be written here to access the data from the database or to perform any operation to the database. But in our implementation we have decided to encapsulate the database and thus the third tier doesn’t have direct access to the database but instead can only access the stored procedures and functions -being the only ones that can access the database thus allowing a more flexible and updatable database.

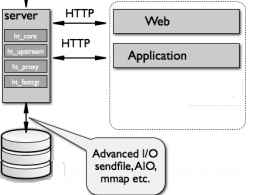


Figure 21 system architecture

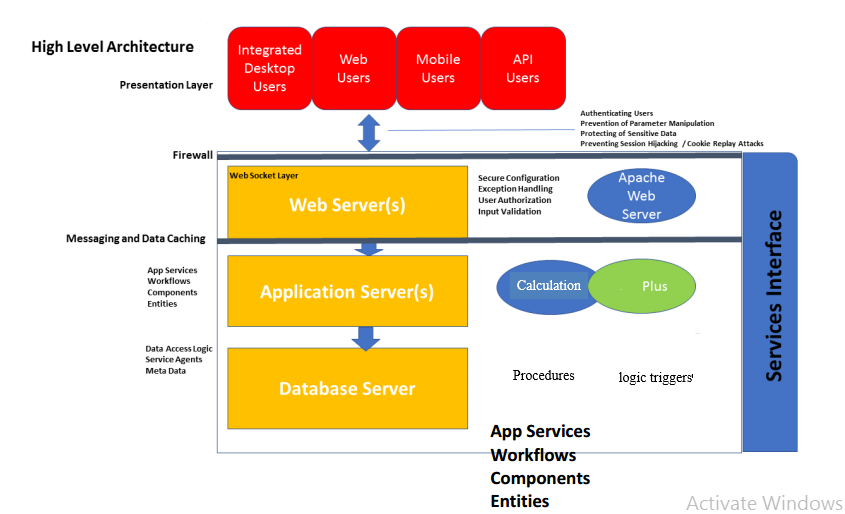
### System process

A perspective on the flow of events might better show the activity performed internally better.



Figure 21 System process

### Request Processor

Figure 21 Sequence diagram for logout

### Subsystem decomposition

In system design it is necessary to reduce the problem into smaller and manageable problems. Or in other words decompose a system into smaller and specific subsystems. And there are lots of ways to go about the decomposition. But for this particular system we have decided to decompose our system using repository style.

Repository style means subsystems do not recognize each other. Or in other words each subsystem acts as if it is the only system and has no interaction with other subsystems. The way how they coordinate is by using a single data point called the repository.

Since our system is mostly data manipulation and thus database eccentric. Repository style models it best. The model is described in the figure below.

Figure 21 Sequence diagram for logout

### Hardware/ software mapping

Hardware/ software mapping shows in detail how the subsystems are implemented. Or shows us in which hardware the subsystem actually executes. By defining their physical locations it also addresses the communication overhead and different process scheduling burdens.

Thus as the figure below shows, there are 3 physical locations for the subsystems to reside.

1. The client side: the interface subsystem resides here and is concerned with Processing the incoming information and presenting it as required by the user
2. The server side: here the mapped processes are those concerned with data management , data processing and data analysis
3. The database side: here only those with direct access to the data reside. In other words the process here are concerned with shielding or encapsulating the database and any other process that wants access to data can only do so by accessing one of this processes.

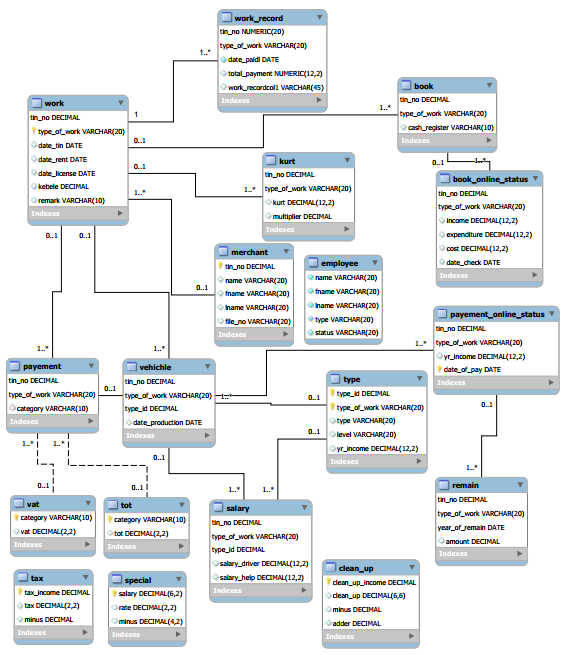
The detailed mapping of each subsystem is identified below:

Figure 21 Sequence diagram for logout

### Component diagram

Figure 21 Sequence diagram for logout

### Database design

Figure 21 Sequence diagram for logout