Software Requirement Specification Document

Al Shams Company for Housing and Construction System

Rowan Omar, Hend Mohamed, Ahmed Hany, Hossam Badr

May 12, 2017

1 Introduction

1.1 Purpose of this document

This document will define an agreement between the client Al shams Company and the developers Team, by formally outlining the functional requirements and non-functional requirements (performance constraints) for the System. These requirements form boundaries and constraints for the projects design process as well as providing a reference for the development of a test process. It will explain system constraints, interface and interactions with other external applications. This document is primarily intended to be proposed to a customer for its approval and a reference for developing the first version of the system for the development team

1.2 Scope of this document

This software system will be a Web Application for a housing and construction company. This system will be designed to maximize the companys productivity by providing tools to assist employees to manage financial and management issues of the company and allow customers to contact the company easily. By maximizing the employees work efficiency and production the system will meet the employees needs while remaining easy to understand and use. More specifically, this system is designed to allow accounting stuff to manage accounting issues such as contracts, transactions, payments, cheques, account statements and letters of guarantee. An administrator also uses the system in order to administer the system and keep the information accurate. It allows the technical stuff to manage time plans, projects plans, cost estimates and demodulators. The software will facilitate communication between customers and the company so, the customer can register for an account and apply for a new project. The system has a user friendly and usable interface, supported by a well designed relational database.

1.3 Overview

The next Section, the Overall Description section, of this document gives an overview of the functionality of the product. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next chapter.
The third Section, Requirements Specification section, of this document is written primarily for the developers and describes in technical terms the details of the functionality of the product.
The fourth Section provides a description of the different system interfaces.
The fifth and sixth Sections provide performance requirements and design constraints.
The seventh Section provides the non-functional attributes.
The eighth Section deals with Preliminary Object-Oriented Domain Analysis which includes Inheritance Relationships and class descriptions.
Last four Section includes Operational Scenarios, Preliminary Schedule Adjusted, Preliminary Budget Adjusted, appendices and references.

1.4 Business Context

The System will save time, effort and will ensure an easy and flexible use. It will be designed to maximize the companys productivity by providing tools to assist employees to manage financial and management issues of the company and it will facilitate communication between customers and the company.

2 General Description

2.1 Product Functions

2.1.1 Admin

1. The system shall allow admin to Add/update/delete/search records such as Projects, Users (Employees, admins and customers), and Accounts.
2. The system shall allow admin to control users accessibility (URLs and User Types permissions).
3. The system shall allow admin to create/update forms of the system.

2.1.2 Technical Office

1. The system shall allow technical office stuff to assign cost estimates, time plans and project plans for different projects.
2. The system shall allow technical office stuff to assign demodulators for

projects.

2.1.3 Accounting Management

1. The system shall allow head of auditing to review and make reports concerning cost estimate, transactions, budgets, account statements and financial positions.
2. The system shall allow head of customer accounts to manage their account statements and create contracts.
3. The system shall allow head of suppliers accounts to manage suppliers account statements.

2.2 Similar System Information

The Builders for Building and constructions Agency.

pictures/Capture.PNG

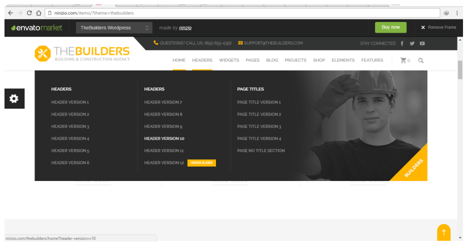


Figure 1: similar system

pictures/Capture1.PNG

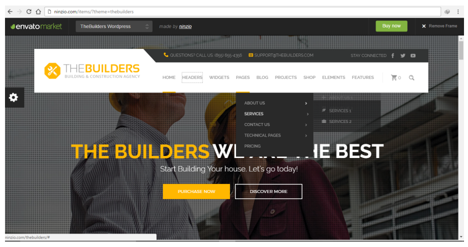

envato market
THE BUILDERS
THE BUILDERS WE ARE THE BEST
Start Building Your house. Let's go today!
PURCHASE NOW
DISCOVER MORE

Figure 2: similar system

2.3 User Characteristics

There are three types of users that interact with the system: customers, employees and administrators. Each of these three types of users has different use of the system so each of them has their own requirements.
1. Customer

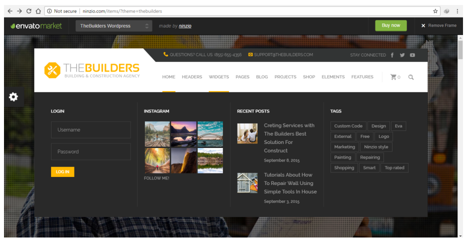
pictures/Capture2.PNG 

Figure 3: similar system

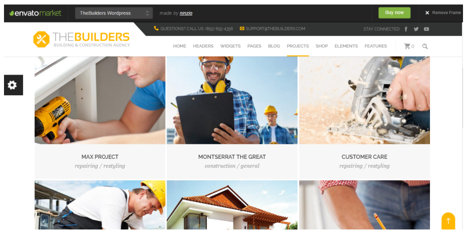
pictures/Capture3.PNG 

Figure 4: similar system

He is expected to be Internet literate and be able to use a search engines. He is expected to be able to use the login system provided in the system in order to apply for his new project.
2. Administrator
He is expected to be able to work with spreadsheets, databases and word processing. He is expected to be able to manage the database through forms provided in the system.
3. Employee
He is expected to have good computing skills. He is expected to have Domain-related knowledge and skills (Accounting knowledge).

2.4 User Problem Statement

This system will save time for the company as they wont need to schedule a lot of meetings with the customers to take their requests for new projects, assign contracts or to see how their projects are going because the system will allow them to see time plans of their projects. The system will also help the company to manage reports and data with no need to have a lot of documents and papers to search in.

2.5 User Objectives

This system will provide an easy use for all employees with a simple interface which helps managing everything related to companys projects easily. It will help them save all data related to the projects and will help them to get any reports that will help them track the progress in their projects with no need to search and analyze too much papers and documents to get this data, Also it will save too much time to get accurate results. The system will also raise customers motivation to request for a new project with no need for them to visit the company and waste time.

2.6 General Constraints

Username and password are used for the identification of users to login.
Only registered customers and employees will be authorized to use the system depending on permissions of each of them.
Limited to HTTP.
This system is working for single server.

3 Functional Requirements

3.1 Back-end Functions

3.2 Administrator API

3.2.1 Function : Add user

code(01)
o Input : (user type, first name, last name, username, email, password, Confirm Password , phone, Address, Country, city)
o Source: UI web form.
o Action: we determine the pages permission according to the user type and generate new user id. Also if a user already exist with either username or email. The admin cannot entered a confirmation password that is not the same as the password. Validate input
o Requirement: the user must be an admin.
o Pre-condition: User email is unique not repeated
o Output: Confirmation of successfully added in database.
o Description: Add different type of employees (ex: Accounting Staff, Cost Controller, Technical Office Staff, Project Manager, Head of the suppliers accounts, Head of the customers accounts, Head of subcontracts accounts, Head of auditing) with auto generated ID.
o Post-condition: the new user will be added in database. User Type Permissions will be applied to the user type
o Dependency: None
o Destination: Return back to Admin control panel.

3.2.2 Function : Add new Project Type

code(02)
o Input: (project type name) with auto generated ID.
o Output: Confirmation of successfully added in database.
o Description: add new type of project with auto generated ID.
o Source: UI web form.
o Action: check if this type exists, the admin will get a warning message. Else, itll be added.
o Requirement: the user must be an admin.
o Pre-condition: project type name must be unique.
o Post-condition: itll be added in the project type list and database.
o Dependency: none.
o Destination: Return back to Admin control panel.

3.2.3 Function: Add URL

code(03)
o Input: URL name , user type
o Output: Confirmation of successfully added in database.
o Description: add URL and determine which type of user has the access on the certain page.
o Source: UI web form.
o Action: check if this URL already exists, the admin will get a warning message. Else, itll be added. Validate input.
o Requirement: the user must be an admin.
o Pre-condition: URL name must be unique.
o Post-condition: itll be added in the URLs and database.
o Dependency: Assign user types to new URL
o Destination: main control loop.

3.2.4 Function: add user type

code(04)
o Input: (user type name) with auto generated ID.
o Output: Confirmation of successfully added in database.
o Description: add new type of users with auto generated ID.
o Source : UI web form.
o Action: check if this type exists, the admin will get a warning message. Else, itll be added.
o Requirement: the user must be an admin.
o Pre-condition: user type name must be unique.
o Post-condition: itll be added in the user type list and database.
o Dependency: Assign URL permissions to the new user type

o Destination: Return back to Admin control panel.

3.2.5 Function: add companys project

code(05)
o Input: (project name, description, array of pictures) with auto generated ID.
o Output: Confirmation of successfully added in database and added in the companys projects page.
o Description: add new companys project with auto generated ID.
o Source: UI web form.
o Action: check if this project exists, the admin will get a warning message. Else, itll be added.
o Requirement: the user must be an admin.
o Pre-condition: project name must be unique.
o Post-condition: itll be added in the projects list and database.
o Dependency : none.
o Destination: go to the companys projects page.

3.2.6 Function : update user

code(06)
o Input: User choice to update (user type, first name, last name, username, email, password, phone, Address, Country, city)
o Output: Confirmation of successfully updated in database.
o Description: update user information in database.
o Source: UI web form.
o Action: update data for the selected radio button.
o Requirement: the user must be an admin to update users information.
o Pre-condition: the user must be in database and the new user name or new Email must be unique.
o Post-condition: itll be updated in database.
o Dependency:if we change user type, you must access new URL permissions.
o Destination: Return to the same update user with the new table record.

3.2.7 Function:update customer

o Input: Customer choice to update (first name, last name, username, email, password, phone, Address, Country, city)
o Output : Confirmation of successfully updated.
o Description : update customer information in database.
o Source: UI web form.
o Action: update data for the selected choice.
o Requirement: the user must be an admin or a customer to update customers

information.
o Pre-condition: the new username or new Email must be unique.
o Post-condition: itll be updated in database.
o Dependency: none.
o Destination: Return to updated user data with the new table record.

3.2.8 Function: edit URL permissions

code(07)
o Input: array of user types thatll take this permission.
o Output: Confirmation of successfully updated in database.
o Description: update user type permissions in database.
o Source: UI web form.
o Action: update data for the selected user types.
o Requirement: the user must be an admin to update users permission.
o Pre-condition: there must be update happened.
o Post-condition: itll be updated in database.
o Dependency: user types permassions will be affected
o Destination: Return back to Admin control panel.

3.2.9 Function: update companys projects

code(08)
o Input: User choice to update (project name, description, array of pictures)
o Output: Confirmation of successfully updated in database.
o Description: Update Companys projects information in database.
o Source: UI web form.
o Action: update data for the selected user choice.
o Requirement: the user must be an admin to update Companys projects information.
o Pre-condition: the Companys project must be in database and the new project name unique.
o Post-condition: itll be updated in database.
o Dependency: none.
o Destination: go to the companys projects page.

3.2.10 Function: delete user

code(09)
o Input: Username or email.
o Output: Confirmation of deleting the user from database.
o Description: delete user from database.
o Source: UI web form.

o Action: delete user data by the user ID.
o Requirement: the user must be an admin to delete users information.
o Pre-condition: the username or the email must be in database.
o Post-condition: itll be updated in database.
o Dependency: the user's data will be deleted with his permissions .
o Destination: Return back to Admin control panel.

3.2.11 Function : delete companys projects

code(10)
o Input: project name.
o Output: Confirmation of deleting the user from database.
o Description: delete Companys projects information in database.
o Source: UI web form.
o Action: delete data for the selected user choice.
o Requirement: the user must be an admin to delete Companys projects information.
o Pre-condition: the Companys project must be in database.
o Post-condition: itll be deleted in database.
o Dependency: none.
o Destination: go to the companys projects page.

3.2.12 Function: delete user type

code(11)
o Input: user type name.
o Output: Confirmation of successfully deleting in database.
o Description: delete user type
o Source: UI web form.
o Action: delete data for the selected user choice.
o Requirement: the user must be an admin.
o Pre-condition: user type name must be in database.
o Post-condition: itll be deleted in the user type list and database.
o Dependency: the user type permissions will be deleted.
o Destination: Return back to Admin control panel.

3.3 Cost Controller API

3.3.1 Function : View Cost Estimation of a specific project

code(12)
o Input: Project ID.
o Output: Cost Estimation for the entered project id.
o Source: UI web form.

o Description: View cost estimation for any project by entering projects id.
o Action: Check that Projects id is found.
o Requirement: User must be from Cost Controller.
o Pre-conditions: Projects id is found.
o Post-conditions: Cost estimation data are fetched from database.
o Destination: Redirect to result page.
o Dependency:None.

3.3.2 Function: Assign projects cost

code(13)
o Input: Project Name, Quantity, Work Statement, Duration (from/to), Price.
o Output: Confirmation of successfully assigning cost.
o Source: UI web form.
o Description: Set the quantity of items that will be performed and amount of money to spend during a specific duration.
o Action: Validate input.
o Requirement: User must be from Cost Controller.
o Pre-condition: Project is in the projects list.
o Post-condition: Cost is set to the project and added in the database.
o Destination: Redirect to projects list.
o Dependency:none.

3.4 Technical Office API

3.4.1 Function: Assign Project Time Plan

code(14)
o Input: Five Stages, Project Name, Project Type, Project Owner, Stage Description, Stage Duration (from/to).
o Output: Confirmation of successfully assigning the time plan.
o Source: UI web form.
o Description: set the time plan for a specific project and setting the five stages; stage one (Drilling and Piling), stage two (Concrete Structure), stage three (Construction Division), stage four (Health and Electricity), stage five (Finishes) with their description and duration.
o Action: Validate input.
o Requirement: User must be from Technical Office.
o Pre-condition: Project is in the projects list.
o Post-condition: Time Plan is set to the project and added in the database.
o Destination: Redirected to the technical office control panel.
o Dependency:None.

3.4.2 Function: Assign Project Plan

code(15)
o Input: Project Manager, Warehouse Manager, Working Site Accountant, Operation Engineer, Number Of Civil Engineers, Number Of Architects, Number Of Electrical Engineers, Number Of Mechanical Engineers, Number Of Drilling Workers, Number Of Concreting Workers (Bases, Walls), Number Of Painting Workers, Number Of Floor Workers, Quantity, Unit, Skills.
o Output: Confirmation of successfully assigning the plan.
o Source: UI web form.
o Description: set the project plan for a specific project, specifying the number of workers needed per each quantity and specifying their skills.
o Action: Validate input.
o Requirement: User must be from technical office.
o Pre-condition: Project is in the projects list.
o Post-condition: Plan is set to the project and added in the database.
o Destination: Redirected to the technical office control panel.
o Dependency:None.

3.4.3 Function: Assign Project Cost Estimate

code(16)
o Input: Project Type, Project Name, Unit, Item Number, Material and Work Release, Unit Price In Number, Unit Price Written, Total Price. o Output: Confirmation of successfully added cost estimate to that project.
o Source: UI web form.
o Description: set the cost estimate for a specific project by choosing the project type and name and entering the quantity of money for each item.
o Action: Validate input.
o Requirement: User must be from technical office.
o Pre-condition: Project is in the projects list.
o Post-condition: Cost is set to the project and added in the database.
o Destination: Redirected to the technical office control panel.
o Dependency:none.

3.4.4 Function: Assign Project Demodulator

code(17)
o Input: Duration (from/to), Contractor Name, Address, Contractor Type, Project Manager, Area Manager, Demodulator Spent, Unit, Previous Work, Current Work, Total, Category, Business Value, Total Business Value, Ratio Incremental, Value Incremental, Net Value, Incremental Net Spent, Down Payment Discount, Benefits, Deductions, Owed to The Disbursement.
o Output: Confirmation of successfully added Demodulator to that project.
o Source: UI web form.

o Description: set the demodulator for a specific project.
o Action: Validate input.
o Requirement: User must be from technical office.
o Pre-condition: Project is in the projects list.
o Post-condition: Demodulator is set to the project and added in the database.
o Destination: Redirected to the technical office control panel.
o Dependency:none.

3.5 Head of suppliers accounts API

3.5.1 Function: Create Account Statement for Suppliers

code(18)
o Input: Code, Account Name, Phone Number, Address, Account Type, Type.
o Output: Confirmation of successfully account statement added to Supplier.
o Source: UI web form.
o Description: set account statement to a supplier.
o Action: Validate input.
o Requirement: User must be the head of suppliers.
o Pre-condition: none.
o Post-condition: account statement is set to specific supplier and saved in database.
o Destination: Redirected to head of suppliers control panel.
o Dependency:

3.5.2 Function: List Account Statements

code(19)
o Input: none.
o Output: All Account Statements for suppliers.
o Source: UI web form.
o Description: List all account statements for all suppliers on the system.
o Action: none.
o Requirement: User must be the head of suppliers.
o Pre-condition: none.
o Post-condition: A list of all suppliers accounts is shown.
o Destination: Redirected to head of suppliers control panel.
o Dependency:

3.5.3 Function: Create Transaction for a Supplier

code(20)
o Input: Transaction Number, Date, Code, Debit, Credit, Account Name, Report.
o Output: Confirmation of successful transaction.
o Source: UI web form.

o Description: make a transaction to/from a specific supplier.
o Action: Validate input.
o Requirement: User must be the head of suppliers.
o Pre-condition: none.
o Post-condition: Transaction is done and saved in the database.
o Destination: Redirected to head of suppliers control panel.
o Dependency:None.

3.6 Head of Customers Accounts API

3.6.1 Function: Create Account Statement for Customers

code(21)
o Input: Code, Account Name, Phone Number, Address, Account Type, Type.
o Output: Confirmation of successfully account statement added to Customer.
o Source: UI web form.
o Description: set account statement to a customer.
o Action: Validate input.
o Requirement: User must be the head of customers accounts.
o Pre-condition: none.
o Post-condition: account statement is set to specific supplier and saved in database.
o Destination: Redirected to head of customers control panel.
o Dependency:None.

3.6.2 Function: List Account Statements

code(22)
o Input: none.
o Output: All Account Statements for customers.
o Source: UI web form.
o Description: List all account statements for all suppliers on the system.
o Action: none.
o Requirement: User must be the head of customers.
o Pre-condition: none.
o Post-condition: A list of all suppliers accounts is shown.
o Destination: Redirected to head of customers control panel.
o Dependency:None.

3.6.3 Function: Create Transaction for a Customer

code(23)
o Input: Transaction Number, Date, Code, Debit, Credit, Account Name, Report.

o Output: Confirmation of successful transaction.
o Source: UI web form.
o Description: make a transaction to/from a specific customer.
o Action: Validate input.
o Requirement: User must be the head the head of customers.
o Pre-condition: none.
o Post-condition: Transaction is done and saved in the database.
o Destination: Redirected to head of customers control panel.
o Dependency:None.

3.7 Head of Sub-Contracts Accounts

3.7.1 Function: Create Account Statement for Sub-Contracts

code(24)
o Input: Code, Account Name, Phone Number, Address, Account Type, Type.
o Output: Confirmation of successfully account statement added to Sub-Contracts.
o Source: UI web form.
o Description: set account statement to a Sub-Contracts.
o Action: Validate input.
o Requirement: User must be the head of Sub-Contracts accounts.
o Pre-condition: none.
o Post-condition: account statement is set to specific Sub-Contracts and saved in database.
o Destination: Redirected to head of Sub-Contracts control panel.
o Dependency:None.

3.7.2 Function: List Account Statements

code(25)
o Input: none.
o Output: All Account Statements for Sub-Contracts.
o Source: UI web form.
o Description: List all account statements for all Sub-Contracts on the system.
o Action: none.
o Requirement: User must be the head of Sub-Contracts.
o Pre-condition: none.
o Post-condition: A list of all Sub-Contracts accounts is shown.
o Destination: Redirected to head of Sub-Contracts control panel.
o Dependency:None.

3.7.3 Function: Create Transaction for a Sub-Contracts

code(26)
o Input: Transaction Number, Date, Code, Debit, Credit, Account Name, Report.
o Output: Confirmation of successful transaction.
o Source: UI web form.
o Description: make a transaction to/from a specific Sub-Contracts.
o Action: Validate input.
o Requirement: User must be the head the head of Sub-Contracts.
o Pre-condition: none.
o Post-condition: Transaction is done and saved in the database.
o Destination: Redirected to head of Sub-Contracts control panel.
o Dependency:None.

3.8 Accounting Stuff API

3.8.1 Function: Calculate Commercial and Industrial Tax

code(27)
o Input: Integrated Works, Extra Items, Insurance, Project Cost.
o Output: Result of Calculations.
o Source: UI web form.
o Description: Subtract Projects cost from taxes to get the net cost of the project.
o Action: Enter data required for calculations. Validate input.
o Requirement: User must be from accounting stuff.
o Pre-condition: none.
o Post-condition: Calculations are shown and saved in database.
o Destination: Redirected to accounting stuff control panel.
o Dependency:None.

3.8.2 Function: List Demodulator

code(28)
o Input: Demodulator Id.
o Output: Demodulator Data.
o Source: UI web form.
o Description: Show a specific Demodulator data by entering id.
o Action: Check that Demodulator Id already exists.
o Requirement: User must be from accounting stuff.
o Pre-condition: Demodulator Id is found.
o Post-condition: Demodulator Data is fetched from database and shown.
o Destination: Redirected to accounting stuff control panel.

o Dependency:None.

3.9 Head of Auditing API

3.9.1 Function: View Transactions (Reports)

code(29)
o Input: Start/End Dates, Account Name and Code.
o Output: All Transactions in a specific Duration.
o Source: UI web form.
o Description: Show a specific Transactions data by entering Duration and account name.
o Action: Check that Account Statement already exists.
o Requirement: User must be head of auditing.
o Pre-condition: Account Statement Code and Name are found.
o Post-condition: Transaction Data is fetched from database and shown.
o Destination: Redirected to head of auditing control panel.
o Dependency: None.

3.9.2 Function: View Cost Estimate (Reports)

code(30)
o Input: Project Name.
o Output: Work Statement and its total price.
o Source: UI web form.
o Description: Show cost estimate for a specific project with all its items.
o Action: Check that Project exists.
o Requirement: User must be head of auditing.
o Pre-condition: Project Name is found.
o Post-condition: Cost Estimate Data is fetched from database and shown.
o Destination: Redirected to head of auditing control panel.
o Dependency: none.

3.9.3 Function: View Financial Position

code(31)
o Input: Duration.
o Output: Financial Position in a specific duration.
o Source: UI web form.
o Description: Show financial position of company in a specific duration.
o Action: none.
o Requirement: User must be head of auditing.
o Pre-condition: none.
o Post-condition: Financial Data are fetched from database, Calculated and

shown.
o Destination: Redirected to head of auditing control panel.
o Dependency: none.

Function-¿ View Companys Budget (Reports):

3.10 Front-end Functions

3.11 Customer API

3.11.1 Function: Register

code(32)
o Input: FirstName,LastName,Email,Address,PhoneNumber,Username,Password,Re-Password
o Output: Confirmation of successful register.
o Description: customer signup for a new account.
o Source: UI web form.
o Action: Check if user already exists. Validate input.
o Requirement: none.
o Pre-condition: User is unique and not in the database.
o Post-condition: User added in the customers list and in database.
o Destination: Redirected to User profile.
o Dependency:None.

3.11.2 Function: Apply For New Project

code(33)
o Input: Land Area, Land Address, Project Type, Project Description, Customer Name, Address, Phone Number.
o Output: Confirmation of successfully submission and a confirmation for the project will be sent.
o Source: UI web form.
o Description: Customer will apply for a new project.
o Source: UI web form.
o Action: Validate input.
o Requirement: User must be a customer.
o Pre-condition: none.
o Post-condition: Project added in the projects list and in database.
o Destination: Redirected to user profile.
o Dependency:None.

4 Interface Requirements

This section provides a detailed description of all inputs into and outputs from the system. It also gives a description of the hardware, software and communication interfaces and provides basic prototypes of the user interface.

4.1 User Interfaces

The user interface for the software shall be compatible to any browser such as Internet Explorer, Mozilla or Netscape Navigator by which user can access to the system.

A first-time customer should see the register form in order to have account on the system, he can also see some information about the company.

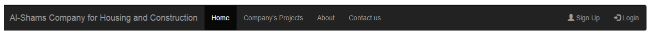
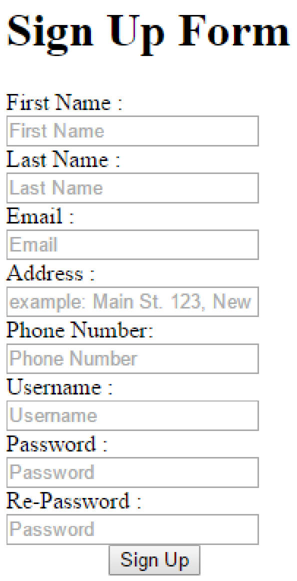
pictures/nav.png 

Figure 5: Home page NAV



Sign Up Form

First Name :
First Name
Last Name :
Last Name
Email :
Email
Address :
example: Main St. 123, New
Phone Number:
Phone Number
Username :
Username
Password :
Password
Re-Password :
Password
Sign Up

pictures/signup.png

Figure 6: Customer Sign Up

If the user is not a first-time customer, he/she should be able to login in order to apply for a new project. If the user has not registered, he/she should be able to do that on the log-in page.

pictures/signin.png

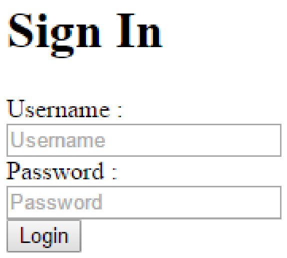


Figure 7: Sign In

If the user is not a first-time customer, he/she should be able to login in order to apply for a new project. If the user has not registered, he/she should be able to do that on the log-in page. pictures/apply project.png

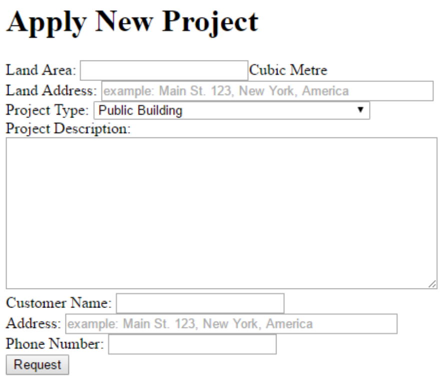


Figure 8: Sign In

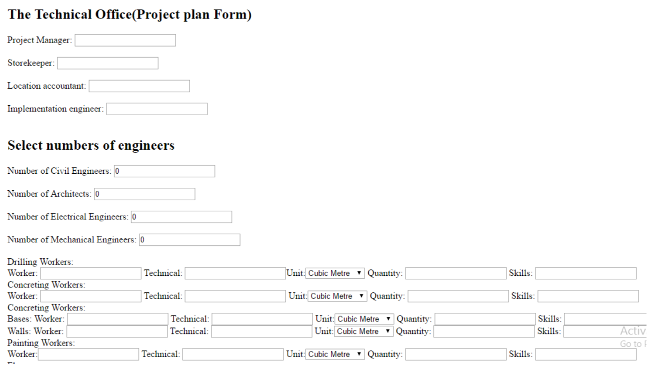
4.2 Hardware Interfaces

Since the application must run over the internet, all the hardware shall require to connect internet will be hardware interface for the system. As for e.g. WAN LAN, Ethernet Cross-Cable.

4.3 Communications Interfaces

The communication between the different parts of the system is important since they depend on each other. However, in what way the communication is achieved is not important for the system and is therefore handled by the underlying operating systems.
Client (customer) on Internet will be using HTTP protocol.
Client (system user) on Internet will be using HTTP protocol.

The company employees and administrators should login in first in order to interact with the system, an employee should be registered on the system in order to log in and manage the company operations depending on his type and the permissions he has. An administrator should also be able to log in to the system where he/she can administer the system by for instance adding, deleting, updating users and companys information. pictures/projectplan.png


The Technical Office(Project plan Form)

Project Manager:

Storekeeper:

Location accountant:

Implementation engineer:

Select numbers of engineers

Number of Civil Engineers: 0

Number of Architects: 0

Number of Electrical Engineers: 0

Number of Mechanical Engineers: 0

Drilling Workers:
Worker: Technical: Unit: Cubic Metre Quantity: Skills:
Concreting Workers:
Worker: Technical: Unit: Cubic Metre Quantity: Skills:
Concreting Workers:
Bases: Worker: Technical: Unit: Cubic Metre Quantity: Skills:
Walls: Worker: Technical: Unit: Cubic Metre Quantity: Skills:
Painting Workers:
Worker: Technical: Unit: Cubic Metre Quantity: Skills:

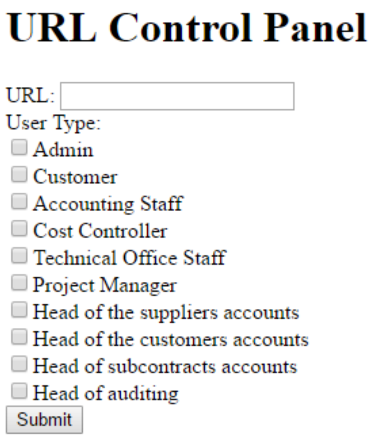
Figure 9: Sign In

4.4 Software Interfaces

Client on Internet
Web Browser, Operating System (any)
Web Server
000webhost, Operating System (any)
Data Base Server
MySQL, Op erating System (any)
Development End
HTML, JAVASCRIPT, AJAX, BOOTSTRAP, MYSQL, PHP, OS (Windows)

5 Performance Requirements

The requirements in this section provide a detailed specification of the user interaction with the software and measurements placed on the system performance.
The product shall be based on web and has to be run from a web server.
The product shall take initial load time depending on internet connection strength which also depends on the media from which the product is run.
The performance shall depend upon hardware components of the client/customer.


URL Control Panel

URL:

User Type:

- ☐ Admin
- ☐ Customer
- ☐ Accounting Staff
- ☐ Cost Controller
- ☐ Technical Office Staff
- ☐ Project Manager
- ☐ Head of the suppliers accounts
- ☐ Head of the customers accounts
- ☐ Head of subcontracts accounts
- ☐ Head of auditing

pictures/Urlcontrolpanel.png

Figure 10: Sign In

5.1 Prominent search feature

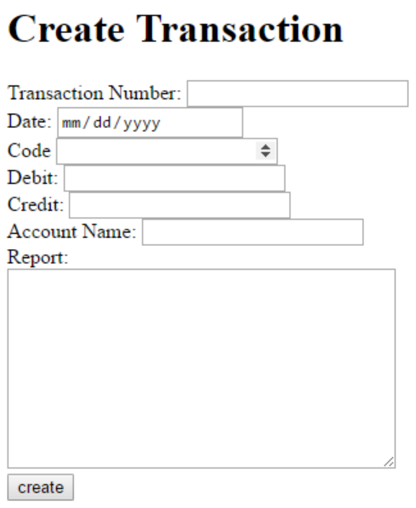
TITLE: Prominent search feature
DESC: The search feature should be prominent and easy to find for the user.
RAT: In order to a user to find the search feature easily.
DEP: none

5.2 Usage of the search feature

TITLE: Usage of the search feature
DESC: The different search options should be evident, simple and easy to understand.
RAT: In order to for a user to perform a search easily.
DEP: none

5.3 Usage of the result in the list view

TITLE: Usage of the result in the list view
DESC: The results displayed in the list view should be user friendly and easy to understand. Selecting an element in the result list should only take one click.
RAT: In order to for a user to use the list view easily.
DEP: none



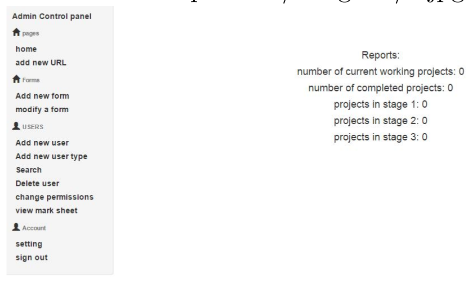
Create Transaction

Transaction Number:
Date: mm/dd/yyyy
Code
Debit:
Credit:
Account Name:
Report:

pictures/transaction.png

Figure 11: Sign In

pictures/imagesss/1.jpg



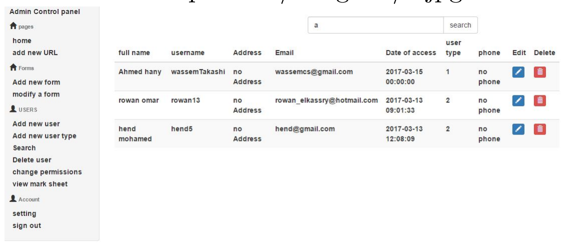
Reports:
number of current working projects: 0
number of completed projects: 0
projects in stage 1: 0
projects in stage 2: 0
projects in stage 3: 0

Figure 12: Sign In

5.4 Usage of the information link

TITLE: Usage of the information link
DESC: The information link should be prominent and it should be evident that it is a usable link.
Selecting the information link should only take one click.
RAT: In order to for a user to use the information link easily.
DEP: none

pictures/imagesss/2.jpg



full name	username	Address	Email	Date of access	user type	phone	Edit	Delete
Ahmed hany	wassemTakashi	no Address	wassemcs@gmail.com	2017-03-15 00:00:00	1	no phone		
rowan omar	rowan13	no Address	rowan_elkassry@hotmail.com	2017-03-13 09:01:33	2	no phone		
hend mohamed	hend5	no Address	hend@gmail.com	2017-03-13 12:08:09	2	no phone		

Figure 13: Sign In

pictures/imagesss/3.jpg

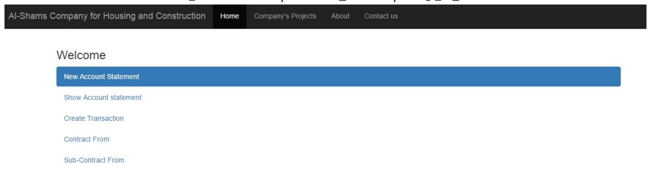


Figure 14: Sign In

5.5 Response time

TAG: Response Time
GIST: The fastness of the search
SCALE: The response time of a search
MUST: No more than 3 seconds 100precent of the time.
WISH: No more than 2 second 100percent of the time.

5.6 System dependability

TAG: System Dependability
GIST: The fault tolerance of the system.
SCALE: If the system loses the connection to the Internet the user should be informed.
MUST: 100percent of the time.

pictures/imagesss/4.jpg

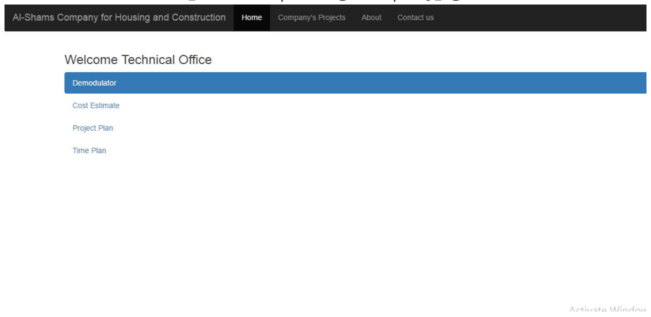


Figure 15: Sign In

pictures/imagesss/7.jpg

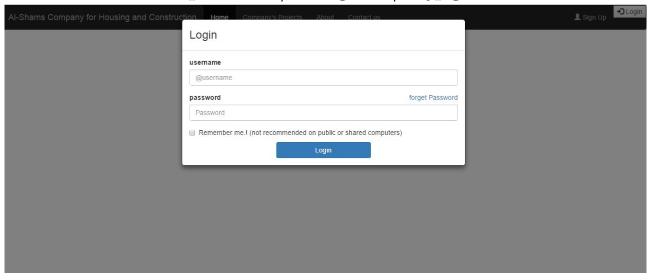


Figure 16: Sign In

6 Design Constraints

This section includes the design constraints on the software caused by the hardware.

6.1 Hard drive space

TAG: Hard Drive Space
GIST: Hard drive space.
SCALE: The applications need of hard drive space.
METER: MB.
MUST: No more than 20 MB.
PLAN: No more than 15 MB.
WISH: No more than 10 MB.
MB: DEFINED: Megabyte

6.2 Application memory usage

TAG: Application Memory Usage
GIST: The amount of Operate System memory occupied by the application.
SCALE: MB.
METER: Observations done from the performance log during testing
MUST: No more than 20 MB.
PLAN: No more than 16 MB
WISH: No more than 10 MB
Operate System: DEFINED: The mobile Operate System which the application is running on.
MB: DEFINED: Megabyte

7 Other non-functional attributes

7.1 Security

TAG: Communication Security
GIST: Security of the communication between the system and server.
SCALE: The messages should be encrypted for log-in communications, so others cannot get user-name and password from those messages.
METER: Attempts to get user-name and password through obtained messages on 1000 log-in session during testing.
MUST: 100percent of the Communication Messages in the communication of a log-in session should be encrypted.
Communication Messages: Defined: Every exchanged of information between client and server.

TAG: Encryption and Decryption
GIST: The security of protecting sensitive data for users of the system.
SCALE: The back-end servers shall never display a users sensitive data such as passwords. It shall always be encrypted with special characters representing typed characters.
METER: Measurements obtained on 1000 hours of usage during testing.
MUST: 100percent of the time.

7.2 Reliability

TAG: System Reliability
GIST: The reliability of the system.
SCALE: The reliability that the system gives the right result on a search.
METER: Measurements obtained from 1000 searches during testing.
MUST: More than 98 percent of the searches.
PLAN: More than 99percent of the searches.

WISH: 100percent of the searches.

7.3 Maintainability

TITLE: Application extendibility
DESC: The application should be easy to extend. The code should be written in a way that it favors implementation of new functions.
RAT: In order for future functions to be implemented easily to the application.
DEP: none

TITLE: Application testability
DESC: Test environments should be built for the application to allow testing of the applications different functions.
RAT: In order to test the application.
DEP: none

8 Preliminary Object-Oriented Domain Analysis

8.1 Inheritance Relationships

8.2 Class descriptions

8.3 User

Abstract class

List of Superclasses:
None.

List of Subclasses:
$Customer, Administrator, cost_controller, Headofsuppliersaccounts, Headofcustomersaccounts, Headofsub$

Purpose:
Main class has users information.

Collaborations:
None.

Attributes:
$user_id : int$
$-first_name : string$
$-last_name : string$
$-gender : string$

$- Address : string$
$- user_type : int$
$- user_name : string$
$- email : string$
$- password : string$
$- phone_number : string$
$- confirmPassword : string$
$- date_of_Access : string$
$- acceptPolicy : int$
$- country : string$
$- city : strin$

Operations
Login (username: string, password: string);
Logout ();

8.3.1 Customer

Abstract.

List of Superclasses:
User

List of Subclasses:
None.

Purpose:
The class has customers data and what they do.

Collaborations:
This class Associates with project class.

Attributes:
The class inherits all attributes from super class

Operations
Register(firstname: string, lastname: string, username: string, email: string, password: string, ConfirmPassword: string, phone: string, Address:string, Country: string, city: string);
$Apply_Project(pro : project)$;

8.3.2 Administrator

Concrete
List of Superclasses:
User

List of Subclasses:
None

Purpose:
The class has customers data and what they do.

Collaborations:
None.

Attributes:
The class inherits all attributes from super class.

Operations:
AddUser(usertype: int, firstname: string, lastname: string, username: string, email: string, password: string, ConfirmPassword: string, phone: string, Address: string, Country: string, city: string) ;
$addProject_Type()$;
$AddURL(url_name : string, User_type : array)$;
$add_user_type(type_name : string)$;
$add_CompanyPronect(project_name : string, description : string, pictures : array)$;
$update_user(userID : string, attribute_ID : string)$;
$edit_URL_Permissions(user_Types : array)$;
$update_company_projects(changes : array)$;
$delete_user_type(userTypeName : int)$;
$delete_Company_project(project_name : string)$;
$delete_User(username : String)$;

8.3.3 Cost Controller

Concrete

List of Superclasses:
User

List of Subclasses:
None.

Purpose:
The class has Cost controller data and what they do.

Collaborations:
This class Associates with IReport Class.

Attributes:
The class inherits all attributes from super class and has other attributes:

startDate: date
- endDate: date
- price: double
$-band_id : int$
$-work_Statement : string$

Operations
$AssignCost(project_Name : string, work_Statement : string, quantity : int, band_ID : int, startDate : date, endDate : date, price : double)$;
$totalPrice(Old_price : double)$;

8.3.4 Technical office

Abstract.

List of Superclasses:
User

List of Subclasses:
None.

Purpose:
The class has technical office data and what they do.

Collaborations:
This class Associates with project plan,timeplan stage,Demodulator,cost Estimate.

Attributes:
The class inherits all attributes from super class.

Operations:
$AssignProjectTimePlan(Timeplan_stage : array[], projectName : string, projectType : string, project_Owener : string)$;
$Assign_Project_plan(PP : project_Plan)$;
$assign_project_CostEstimate(CE : cost_Estimate)$;
$Assign_Project_Demodulator(DMO : Demodulator)$;

8.3.5 Head of supplires Accounts

Abstract.

List of Superclasses:
User

List of Subclasses:
None.

Purpose:
The class has Head of supplires Accounts data and what they do.

Collaborations:
This class Associates with transaction and Account statement.

Attributes:
The class inherits all attributes from super class.

Operations:
None.

8.3.6 Head of customer Accounts

Abstract

List of Superclasses:
User

List of Subclasses:
None.

Purpose:
The class has Head of customer Accounts data and what they do.

Collaborations:
This class Associates with transaction , Account statement and contract.

Attributes:
The class inherits all attributes from super class.

Operations:
createContract( ) ;

8.3.7 Head of subcontract Accounts

Abstract.

List of Superclasses:
User

List of Subclasses:
None.

Purpose:
The class has Head of subcontract Accounts data and what they do.

Collaborations:
This class Associates with transaction , Account statement and subcontract.

Attributes:
The class inherits all attributes from super class.

Operations:
createSubcontract( ) ;

8.3.8 Accounting stuff

Abstract.

List of Superclasses:
User

List of Subclasses:
Nonne.

Purpose:
The class has Accounting stuff data and what they do.

Collaborations:
This class Associates with taxes.

Attributes:
The class inherits all attributes from super class.

Operations:
$calculate_Commercial_And_Industrial_Tax();$

8.3.9 Project plan

Abstract.

List of Superclasses:
None.

List of Subclasses:
None.

Purpose:
The class has project plan data and what they do.

Collaborations:
This class aggregated with Worker needs.

Attributes:
$civil_Engineer_no : int$
$-Architects_no : int$
$-Electrical_Engineers_no : int$
$-Mechanical_Engineers_no : int$
$-project_Manager : string$
$-warehouse_Manager : string$
$-Operation_Eng : string$
$-Worksite_Accountant : string$
$-worker_needs : Array[]$

Operations:
None.

8.3.10 Worker needs

Abstract.

List of Superclasses:
None.

List of Subclasses:
None.

Purpose:
The class has worker needs to each project data and what they do.

Collaborations:
This class aggregated with unit.

Attributes:
$-unit_name : string$
$-price_Written : string$
$-price_no : double$

Operations:
None.

8.3.11 Timeplan stage

Abstract.

List of Superclasses:
None.

List of Subclasses:
None.

Purpose:
The class has time plan to each project Data.

Collaborations:
None.

Attributes:
- From: date
- to: date
- Description: string
- name: string
Operations:
None.

8.3.12 Cost Estimate

Abstract.
List of Superclasses:
None.

List of Subclasses:
None.

Purpose:
The class has cost Estimate to each project data and what they do.

Collaborations:
This class aggregated with project and item.

Attributes:
- id: int
- project: project
- Item: Array[]

Operations:
None.

8.3.13 project

Abstract.

List of Superclasses:
None.

List of Subclasses:
None.

Purpose:
The class has projects data and what they do.

Collaborations:
None.

Attributes:
$-Land_Area : double$
- id: int
$-project_id : int$
$-land_Address : string$
$-project_Type : string$
$-project_Description : string$
- C: customer
- projectName: string
Operations:
None.

8.3.14 Demodulator

Abstract.

List of Superclasses:
None.

List of Subclasses:
None.

Purpose:
The class has Demodulator data and what they do.

Collaborations:
This class aggregated with Demodulator info, incremental Net and unit .

Attributes:
- id: int
- Spent: double
- unit: unit
$-previous_work : doube$
$-current_work : double$
- total: double
- Category: double
$-Business_value : double$
$-total_Bussiness_Value : double$
$-Ratio_incrementals : double$
$-Value_incrementals : double$
$-Net_Value : double$
$-IN : Incremental_Net$
Operations:
$calculate_Total_Work(previousWork : double, CurrentWork : double)$;

8.3.15 Demodulator info

Abstract.

List of Superclasses:
None

List of Subclasses:
None.

Purpose:
The class has demodulator information.

Collaborations:
This class aggregated with Demodulator.

Attributes:
- id: int
- from: date
- to: date

- contractor: string
- Address: string
$-Contractor_Type : string$
$-Project_Manager : string$
$-Area_Manager : string$
Operations:
None.

8.3.16 Incremental Net

Abstract.

List of Superclasses:
None.

List of Subclasses:
None.

Purpose:
The class has part of demodulator information.
Collaborations:
This class aggregated with Demodulator.

Attributes:
- id: int
- spent: double
$-Downpayment_Discount : double$
- Benefits: double
- Deductions: double
$-Owed_to_the_disbursement : double$
Operations:
None.

8.3.17 Contract

Abstract.

List of Superclasses:
None.

List of Subclasses:
subcontract

Purpose:

The class has contract information.

Collaborations:
This class aggregated with pricingData and Tender Planned Distribution.

Attributes:
- id: int
$-Owner_Address : string$
$-Contract_Number : string$
$-notbook_no : int$
- startDate: date
$-State_of_contract : string$
$-Initial_Receipt : date$
$-Final_Receipt : date$
$-Total_Transaction_Value : double$
$-Downpayment_Value : double$
$-Downpayment_Ratio : double$
$-Contract_Type : string$
$-Tender_Data : Tender_Planned_Distribution$
- PD: pricingData
Operations:
None.

8.3.18 subcontract

Abstract.

List of Superclasses:
contract.

List of Subclasses:
None.

Purpose:
The class has subcontract information.

Collaborations:
This class aggregated with Tender Planned Distribution.

Attributes:
$-Editorial_date : date$
- id: int
$-primary_recieving_date : date$
$-Total_Contract_value : double$
$-contract_status : string$
$-final_Recieving_date : date$

Operations:
None.

8.3.19 Tender Planned Distribution

Abstract.

List of Superclasses:
None.

List of Subclasses:
None.

Purpose:
The class has Tender Planned Distribution information.

Collaborations:
This class aggregated with contract and subcontract.

Attributes:
$-tender_{n}otebook_{n}o : int$
$-tender_{i}d : int$
$-tender_{D}escription : string$
- Owner: string
$-No_{p}lanned_{d}istribution : int$
$-No_{N}otebook_{D}istribution : int$

Operations:
None.

8.3.20 pricingData

Abstract.

List of Superclasses:
None.

List of Subclasses:
None.

Purpose:
The class has pricingData information.

Collaborations:
This class aggregated with contract and Price Offering Data.

Attributes:
$-Pricing_no : int$
$-Pricing_Case : string$
$-Notebook_Ppricing_No : int$
$-POD : price_Offering_Data$
Operations:
None.

8.3.21 Tax

Abstract.

List of Superclasses:
None.

List of Subclasses:
None.

Purpose:
The class has Taxes information.

Collaborations:
This class associated with Accounting stuff.

Attributes:
$-Integrated_Works : double$
$-Extra_Items : double$
- insurance: double
Operations:
None.

8.3.22 Head of auditing

Abstract.

List of Superclasses:
User.

List of Subclasses:
None.

Purpose:
The class has demodulator information.

Collaborations:
This class aggregated with Demodulator.

Attributes:
The class inherits all attributes from super class.

Operations:
ViewReport();

8.3.23 IHead

Interface class

List of Superclasses:
None.

List of Subclasses:
None.

Purpose:
Its interface class .
Collaborations:
$TheclassinteractwithCustomer, Administrator, cost_{c}ontroller, Headofsuppliersaccounts, Headofcustomers$
$, Headofsubcontractsaccounts, Accountingstuff, Headofauditing.$

Attributes:
None.

Operations:
$Create_{A}ccount_{S}tatement() : void$
$show_{A}ccount_{S}tatements(code : int) : void$
$Create_{T}ransaction() : void$

8.3.24 IReport

Interface class.

List of Superclasses:
None.

List of Subclasses:
None.

Purpose:
Its interface class.

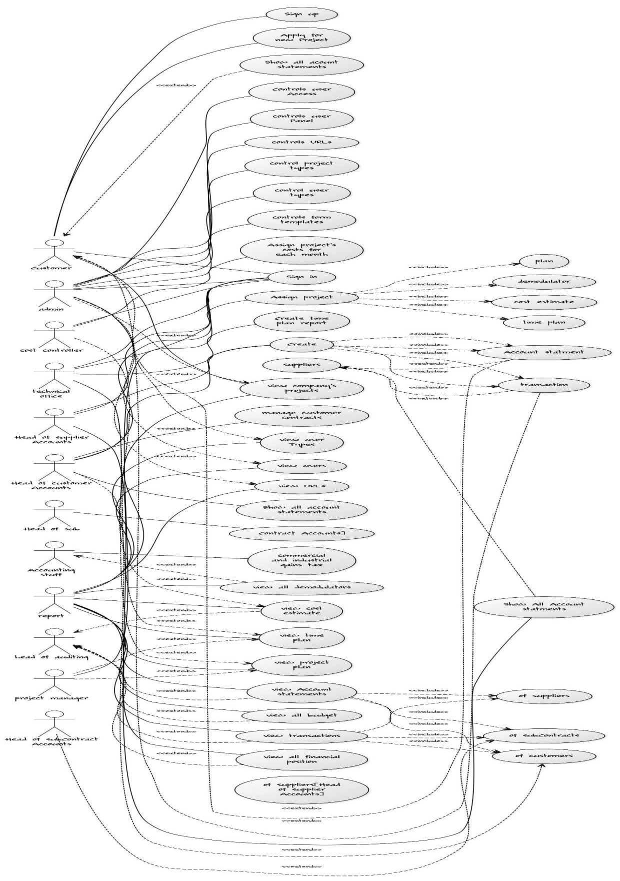
Collaborations:

$TheclassinteractwithCustomer, Administrator, cost_{c}ontroller, Headofsuppliersaccounts$
$, Headofcustomersaccounts, Headofsubcontractsaccounts, Accountingstuff, Headofauditing.$

Attributes:
None.

Operations:
viewReport(ReportType: string) : void

9 Operational Scenarios



pictures/oS.jpg

Figure 17: USE CASE

9.1 Scenario for managing project:

The customer will login in using his user name and password then he will be redirected to apply for a new project form. The customer will enter the data needed to fill the form and will submit it.
The technical office stuff after recieving the project will do the following operations on the project first, the stuff will assign cost estimate determining items needed with their costs then they will assign the time plan including the stages that the project will go through. Finally, They will assign the project plan including all the architects, workers needed for quantity of work with their skills. The technical office stuff will get each month a report including the amount of work done in each project and the amount of money spent on it.

10 Appendices

10.1 Definitions, Acronyms, Abbreviations

User: Someone who interacts with the mobile phone application.
Admin/Administrator: System administrator who is given specific permission for managing and controlling the system.
Stakeholder: Any person who has interaction with the system who is not a developer.
Employee: Any person who has interaction with the system in the company.(Admins, Accounting stuff, Technical office stuff, cost controllers, projects managers, Heads of the suppliers, customers and subcontracts accounts.
Customers: The owners of projects who interact with the system.
System: The web application which present special facilities for the company.
DESC: Description
RAT: Rational
DEP: Dependency TAG: A unique, persistent identifier contained in a PLanguage statement [2].
GIST: A short, simple description of the concept contained in a PLanguage statement [2].
SCALE : The scale of measure used by the requirement contained in a PLanguage statement [2].
METER: The process or device used to establish location on a SCALE contained in a PLanguage statement [2].
MUST: The minimum level required to avoid failure contained in a PLanguage statement [2].
PLAN: The level at which good success can be claimed contained in a PLanguage statement [2].
WISH: A desirable level of achievement that may not be attainable through available means contained in a PLanguage statement [2].
HTTP: Hypertext Transfer Protocol. Its a service protocol.
HTML: HTML (Hypertext Markup Language) is the set of markup symbols or codes inserted in a file intended for display on a World Wide Web browser page.

PHP: PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.
AJAX: AJAX stands for Asynchronous JavaScript and XML. In a nutshell, it is the use of the XMLHttpRequest object to communicate with server-side scripts.
MySQL: MySQL is an open source relational database management system. Information in a MySQL database is stored in the form of related tables.

11 References

Software Engineering 9th edition, Sommervillie, Addison Wesle
http://ninzio.com/items/?theme=thebuilders