**Design Document**

**Faculty Purchase Orders Page**

**Team 14**

Richard Rogers, Hector Roldan, Aaron Sandow, Gunnar Skotnicki

**Three Tier Representations:**

**Client Side:**

The client will interact with the site through their web browser. The technologies present at this level are the HTML, CSS, and PHP to establish a look and feel for the site. PHP is used to put out HTML with the dynamic information from the database. However all the information is gather and send from the server. PHP will create a cookie containing your session ID, this ID relates to the session on the server that contains information such as account type (admin/accountant/purchaser).

**Server Side:**

The server is a Linux VPS (Ubuntu 12.04v) with PHP, Apache, JRE/JDK, and mySQL server. The server contains all the necessary pages for the website to function as well as containing all of the scripts needed for navigating the site. At this level the PHP scripts and Jar file will run. When requested by a user the appropriate file will run returning information to the user. A complete example is this - the admin wants to view all users, selecting that option in their web browser, this sends a POST request to the PHP script that verifies that the user is an admin (through the cookie) and will also accept what type of query the user is doing. Once the PHP recognizes what type of query it is it will then pass the necessary parameters to the Jar file where the database is queried and the results are returned as a JSON arrays for any queries that require information back, or it will return a value to signify that the query type was just to update elements in the database. These returns are sent back to the PHP file, which will decode the JSON array into a PHP array, that we can send to another PHP file to output the queried data to the web browser. The session variables used by PHP correspond to a users' cookie and this session will allow the system to determine what the user can and cannot do.

**Database Level:**

The mySQL server contains a single database (in our case named 'mydb'), and this database contains the stored procedures and functions required for this project. Currently we have 27 stored procedures and 1 stored function. We use a single jar file to handle all of the access to the database and handle queries and calls accordingly.

The database has two tables within one for the Users and another for the Orders.

-The Users table follows the schema of **USERS(userID, username, password, firstName, lastName, emailAddress, role)** where we use emailAddress as the primary key, since no two users should have the same email. All of the values within this table are encrypted with the exception of emailAddress and role. The username and password are encrypted with mySQL's SHA1() command. While userID, firstName, and lastName are encrypted with mySQL's AES\_ENCRYPT() function using the admin's password to encrypt this information.

-The Orders table uses the schema of **ORDERS(orderID, userID, orderRequestDate, purchaseDate, approvalDate, receiveDate, accountNumber, urgent, computerPurchase, vendor, itemDesc, preOrderNotes, attachment, requestor, requestorEmail, Amount, accountCode, PONumber, postOrderNotes)** where we use orderID as the primary key for that table. None of the values of this table are explicitly encrypted.

|  |
| --- |
| **The MySQL statement for creating the Orders Table**  CREATE TABLE IF NOT EXISTS `mydb`.`Orders` (  `OrderId` VARCHAR(40) NOT NULL,  `OrderReqDate` DATE NULL,  `PurchaseDate` DATE NULL,  `ApprovalDate` DATE NULL,  `ReceiveDate` DATE NULL,  `AcctNumber` VARCHAR(100) NULL,  `Urgent` TINYINT(1) NULL DEFAULT 0,  `CompPurchase` TINYINT(1) NULL DEFAULT 0,  `Vendor` VARCHAR(100) NULL,  `ItemDesc` VARCHAR(300) NULL,  `preOrderNotes` VARCHAR(500) NULL,  `Attachment` VARCHAR(150) NULL,  `Requestor` VARCHAR(100) NULL,  `ReqEmail` VARCHAR(150) NULL,  `Amount` DOUBLE NULL,  `AcctCode` VARCHAR(100) NULL,  `PONumber` VARCHAR(100) NULL,  `PostOrderNotes` VARCHAR(500) NULL,  PRIMARY KEY (`OrderId`))  ENGINE = InnoDB |
| **The MySQL statement for creating the Users Table**  CREATE TABLE IF NOT EXISTS `mydb`.`Users` (  `UserID` VARCHAR(100) NOT NULL,  `Username` VARCHAR(100) NULL,  `Pass` VARCHAR(100) NULL,  `FirstName` VARCHAR(100) NULL,  `LastName` VARCHAR(100) NULL,  `Email` VARCHAR(150) NULL,  `Role` VARCHAR(100) NULL,  PRIMARY KEY (`UserID`),  UNIQUE INDEX `Username\_UNIQUE` (`Username` ASC),  UNIQUE INDEX `Email\_UNIQUE` (`Email` ASC))  ENGINE = InnoDB |
| **The MySQL Stored Procedures:** |
| CREATE PROCEDURE `AcctAllApproved` ()  BEGIN  SELECT \*  FROM Orders  WHERE ApprovalDate != NULL  ORDER BY Urgent DESC;  END |
| CREATE PROCEDURE `AcctAllOrders` ()  BEGIN  SELECT \*  FROM Orders  ORDER BY Urgent DESC;  END |
| CREATE PROCEDURE `AcctAwaitingApproval` ()  BEGIN  SELECT \*  FROM Orders  WHERE ApprovalDate = NULL  ORDER BY Urgent DESC;  END |
| CREATE PROCEDURE `AcctOrdersByDateRange` (  IN BeginDate DATE, IN EndDate DATE  )  BEGIN  SELECT \*  FROM Orders  WHERE OrderReqDate >= BeginDate AND OrderReqDate <= EndDate  ORDER BY Urgent DESC;  END |
| CREATE PROCEDURE `AddOrder`  (  IN OID VARCHAR(40), IN AcctNum VARCHAR(100), IN Urg boolean, IN CompPur Boolean, IN Vend VARCHAR(100), IN Descrip VARCHAR(300),  IN Attach VARCHAR(150), IN Req VARCHAR(100), IN ReqEm VARCHAR(150), IN Amt DOUBLE, IN AcctC VARCHAR(100), IN PONum VARCHAR(100)  )  BEGIN  INSERT INTO Orders(  OrderId, OrderReqDate, PurchaseDate, ApprovalDate, ReceiveDate, AcctNumber, Urgent, CompPurchase, Vendor, ItemDesc, PreOrderNotes, Attachment, Requestor,  ReqEmail, Amount, AcctCode, PONumber, PostOrderNotes  )  VALUES(  OID, CURDATE(), NULL, NULL, NULL, AcctNum, Urg, CompPur, Vend, Descrip, NULL, Attach, Req, ReqEm, Amt, AcctC, PONum, NULL);  END |
| CREATE PROCEDURE `AddUser` (  IN UN varchar(100), IN PW varchar(100), IN FN varchar(100), IN LN varchar(100), IN EmailVal varchar(150), IN RoleVal varchar(100), IN eKey varchar(100)  )    BEGIN  INSERT INTO Users(  UserID, Username, Pass, FirstName, LastName, Email, Role  )  VALUES  (  SHA1(UN), AES\_ENCRYPT(UN, eKey), SHA1(PW), AES\_ENCRYPT(FN, eKey), AES\_ENCRYPT(LN, eKey), EmailVal, RoleVal  );  END |
| BEGIN  SELECT OrderId, Requestor, ReqEmail, ItemDesc, Vendor, Amount, Urgent, Attachment  FROM Orders  WHERE ApprovalDate != null  ORDER BY Urgent DESC;  END |
| CREATE PROCEDURE `AllOrders` ()  BEGIN  SELECT OrderId, Requestor, ReqEmail, ItemDesc, Vendor, Amount, Urgent, Attachment  FROM Orders  ORDER BY Urgent DESC;  END |
| CREATE PROCEDURE `AllPurchased` ()  BEGIN  SELECT OrderId, Requestor, ReqEmail, ItemDesc, Vendor, Amount, Urgent, Attachment  FROM Orders  WHERE PurchaseDate != null  ORDER BY Urgent DESC;  END |
| BEGIN  SELECT OrderId, Requestor, ReqEmail, ItemDesc, Vendor, Amount, Urgent, Attachment  FROM Orders  WHERE ApprovalDate = null  ORDER BY Urgent DESC;  END |
| BEGIN  SELECT OrderId, Requestor, ReqEmail, ItemDesc, Vendor, Amount, Urgent, Attachment  FROM Orders  WHERE PurchaseDate != null AND ReceiveDate = null  ORDER BY Urgent DESC;  END |
| BEGIN  SELECT OrderId, Requestor, ReqEmail, ItemDesc, Vendor, Amount, Urgent, Attachment  FROM Orders  WHERE PurchaseDate = null  ORDER BY Urgent DESC;  END |
| CREATE PROCEDURE `ChangePassword` (IN uName VARCHAR(100), IN newPass VARCHAR(100))  BEGIN  UPDATE Users  SET Pass = SHA1(newPass)  WHERE UserId = SHA1(uName);  END |
| CREATE PROCEDURE `EditOrder` (  IN OID VARCHAR(40), IN ORD DATE, IN PDate DATE, IN ADate DATE, IN RDate DATE, IN AcctNum VARCHAR(100), IN Urg boolean, IN CompPur Boolean, IN Vend VARCHAR(100), IN Descrip VARCHAR(300),  IN PrON VARCHAR(500), IN Attach VARCHAR(150), IN Req VARCHAR(100), IN ReqEm VARCHAR(150), IN Amt DOUBLE, IN AcctC VARCHAR(100), IN PONum VARCHAR(100), IN PoON VARCHAR(500)  )  BEGIN  UPDATE Orders  SET OrderReqDate = if (ORD IS null, OrderReqDate, ORD) , PurchaseDate = if(PDate IS null, PurchaseDate, PDate), ApprovalDate = if(ADate IS null, ApprovalDate, ADate), ReceiveDate = if(RDate IS null, ReceiveDate, RDate), AcctNumber = if(AcctNum IS null, AcctNumber, AcctNum), Urgent = if(Urg IS null, Urgent, Urg), CompPurchase = if(CompPur IS null, CompPurchase, CompPur), Vendor = if(Vend IS null, Vendor, Vend), ItemDesc = if(Descrip IS null, ItemDesc, Descrip),  PreOrderNotes = if(PrON IS null, PreOrderNotes, PrON), Attachment = if(Attach IS null, Attachment, Attach), Requestor = if(Req IS null, Requestor, Req), ReqEmail = if(ReqEm IS null, ReqEmail, ReqEm), Amount = if(Amt IS null, Amount, Amt), AcctCode = if(AcctC IS null, AcctCode, AcctC), PONumber = if(PONum IS null, PONumber, PONum), PostOrderNotes = if(PoON IS null, PostOrderNotes, PoON)  WHERE OrderId = OID;  END |
| CREATE PROCEDURE `EditUser` (  IN UN VARCHAR(100), IN FN VARCHAR(100), IN LN VARCHAR(100), IN emailAddr VARCHAR(150), IN rol VARCHAR(100), IN eKey VARCHAR(100)  )  BEGIN  UPDATE Users  SET FirstName = AES\_ENCRYPT(FN, eKey), LastName = AES\_ENCRYPT(LN, eKey), Email = emailAddr, Role = rol  WHERE SHA1(UN) = UserID;  END |
| CREATE PROCEDURE `FindOrderByEmail` (IN emailaddr VARCHAR(150))  BEGIN  SELECT Requestor, ReqEmail, ItemDesc, Vendor, Amount, Urgent, Attachment  FROM Orders  WHERE ReqEmail = emailaddr;  END |
| CREATE PROCEDURE `FindOrderById` (IN ordernum VARCHAR(40))  BEGIN  SELECT Requestor, ReqEmail, ItemDesc, Vendor, Amount, Urgent, Attachment  FROM Orders  WHERE OrderId = ordernum;  END |
| CREATE PROCEDURE `FindOrderByName` (IN reqname VARCHAR(100))  BEGIN  SELECT Requestor, ReqEmail, ItemDesc, Vendor, Amount, Urgent, Attachment  FROM Orders  WHERE Requester = reqname;  END |
| CREATE PROCEDURE `FindOrderByPartName` (IN reqname VARCHAR(100))  BEGIN  SELECT Requestor, ReqEmail, ItemDesc, Vendor, Amount, Urgent, Attachment  FROM Orders  WHERE Requestor LIKE '%' || reqname || '%';  END |
| CREATE PROCEDURE `ForgotPassword` (IN userEmail VARCHAR(150), IN newPass VARCHAR(100))  BEGIN  UPDATE Users  SET Pass = SHA1(newPass)  WHERE Email = userEmail;  END |
| CREATE PROCEDURE `GetEmail` (IN UN VARCHAR(100))  BEGIN  SELECT Email  FROM Users  WHERE SHA1(UN) = UserID;  END |
| CREATE PROCEDURE `GetRoleByEmail` (IN userEmail VARCHAR(150))  BEGIN  SELECT Role  FROM Users  WHERE Email = userEmail;  END |
| CREATE PROCEDURE `OrdersByDateRange` (IN beginDate DATE, IN endDate DATE)  BEGIN  SELECT Requestor, ReqEmail, ItemDesc, Vendor, Amount, Urgent, Attachment  FROM Orders  WHERE OrderReqDate >= beginDate AND OrderReqDate <= endDate  ORDER BY Urgent DESC;  END |
| CREATE PROCEDURE `ReEncrypt` (  IN OldKey VARCHAR(100), IN NewKey VARCHAR(100)  )  BEGIN  UPDATE Users  SET Username = AES\_ENCRYPT(AES\_DECRYPT(Username, OldKey), NewKey), FirstName = AES\_ENCRYPT(AES\_DECRYPT(FirstName, OldKey), NewKey), LastName = AES\_ENCRYPT(AES\_DECRYPT(LastName, OldKey), NewKey);  END |
| CREATE PROCEDURE `RemoveOrder`  (  IN OID VARCHAR(40)  )  BEGIN  DELETE  FROM Orders  WHERE OrderId = OID;  END |
| CREATE PROCEDURE `RemoveUser` (  IN UN VARCHAR(100)  )  BEGIN  DELETE  FROM Users  WHERE UserID = SHA1(UN);  END |
| CREATE FUNCTION `VerifyPassword` (UN VARCHAR(100),PW VARCHAR(100))  RETURNS VARCHAR(45)  BEGIN  DECLARE verified boolean DEFAULT 0;  DECLARE storedPW VARCHAR(100) DEFAULT null;  DECLARE roleVal VARCHAR(45) DEFAULT "-1";  SET storedPW = (SELECT Pass FROM Users WHERE SHA1(UN) = UserID);  IF storedPW = SHA1(PW) THEN SET verified = 1;  ELSE SET verified = 0;  END IF;  IF verified = 1  THEN SET roleVal = (SELECT Role FROM Users WHERE SHA1(UN) = UserID);  ELSE SET roleVal = "-1";  END IF;  RETURN roleVal;  END |
| CREATE PROCEDURE `ViewUsers` (IN adminPass VARCHAR(100))  BEGIN  SELECT UserID, AES\_DECRYPT(Username, adminPass), AES\_DECRYPT(FirstName, adminPass), AES\_DECRYPT(LastName, adminPass), Email, Role  FROM Users;  END |

**ER Diagram on the next page.**

