ADD CONSULTING INC.

SOFTWARE DESIGN DOCUMENT

ONLINE WORK ORDER REQUEST SYSTEM

Asif Baksh

TwiSTedFuSion@aol.com

Denny Singh

suprazrule@yahoo.com

Damian Z Shameer

Yankeey2g@msn.com

Software Design Document v2

TABLE OF CONTENTS

SOFTWARE DESIGN DOCUMENT

ONLINE WORK ORDER REQUEST SYSTEM

PROJECT ABSTRACT

The O.L.W.O.R System will allow residents living in Pace NYC residential halls to place work order request via an online application. This online system will make the system of submitting work orders easy and simple for residents living on campus. Campus Housing Staff members will be able to use the online system to track and manage all incoming work order request. The new system will provide a simple and efficient way for residents and staff members to handle work order requests.

DISTRIBUTION NOTES:

Reviewer List:
C Scharff, Software Engineering, Pace University

Distribution List:
Michelle Perez, Campus Housing, Director of Housing

Diane White, Campus Housing Operations

DOCUMENT REVISION HISTORY

File Name		OLWOR System	
Revision	Date	Author(s)	Revision Notes
1.0	04/05/04	Denny Sign Asif Baksh Zulfikar (Damian) Shameer	Initial draft version
2.0	05/05/04	Zulfikar (Damian) Shameer	Second Review

PURPOSE

This document describes a functional design solution to a particular problem as identified by a requirements specification. This document, the design specification, is a living document that reflects changes to the project. These changes generally take the form of issues that are unclear initially but become resolved after further research or experience (use the *Issues* section as appropriate).

SHORT DESCRIPTION

The world is evolving everyday. Technology is constantly changing, allowing systems to be created with powerful and robust functionalities. The O.L.W.O.R System will seek to take advantage of these emerging technologies, especially web-based technologies, to assist improve processes and procedures by Pace's Campus Housing. With the O.L.W.O.R System, Campus Housing will have the ability to eliminate a major portion of their paper-based and time consuming processes by taking advantage of automated procedures, easy work order requests management, and useful reporting.

Some of the main functions of the O.L.W.O.R System include-

- Online, updatable profiles for residents
- Resident and Staff's ability to submit work orders request online

- Staff can update work order request status
- Staff can manage residential halls
- Director of Housing can grant and revoke access to the system

TARGET AUDIENCE

This document is used by a variety of entities. Certainly, the key audience is Development. However, Marketing, Support, and Training will also refer to this document both to guide their activities as well as to check that all is in accordance with their understanding of the project as a whole. Only rarely are customers directly involved with this document. Customers generally are asked to review the methodology specification, which has some dependence on this document.

SPECIFICATION FLOW

The design specification is finalized after the requirements specification is approved and should generally be developed and reviewed in parallel with the methodology specification to ensure the two are in synch. The design specification is the core document for the project and is always written. A particularly large development effort may have a set of functional specifications.

INTRODUCTION - REQUIREMENTS

The idea behind the development of the Online Work Order Request System is to provide a solution to the difficulty of submitting work orders. Currently, residents must fill out written forms and hand them in to campus housing managers without any work order updates or completion status. Furthermore, there isn't much of a procedure in place to submit work orders. Since the resident is unaware of the progress of a submitted work order, it is not feasible or efficient to submit work order request manually. Another issue associated with manually submitting work orders is that the process is time consuming and tedious, for campus housing managers must keep track of all paper work orders. The development of Online Work Order Request System is in the hope that it will provide a solution for the following requirements:

1. The resident should be able to submit work orders quickly: currently, resident must fill out paper form or speak to CHO Staff directly.
2. The resident should be able to change, update, and/or modify his or her profile which is required to use the system.

3. The resident should be able to view past work orders.
4. CHO Staff should be able to view work orders, update status of submitted work orders, and forward work orders to specified department(s).
5. DOH administration should be able to do all of the functionality of CHO Staff as well as add, remove, and edit, users (CHO Staff included) of the system.

Also, the Online Work Order Request System should provide solutions to the requirements that existing products satisfy:

6. The user should be able to access web pages with the browser frame: All browser products provide the functionality of being able to access web pages from the World Wide Web (WWW).

INTRODUCTION - SOLUTION

The fundamental solution to the problem is to provide the resident with the ability to login to the system with a user name and password to submit work orders. CHO Staff and DOH would have the ability to efficiently keep track of submitted work orders and update the progress of each; thereby, providing a fast and efficient process for submitting work orders and obtaining a response. The current process does not allow the resident or managers to retrieve completion status of submitted work orders. By providing a database for submitted work orders, the resident, CHO Staff, and DOH can simultaneously access the system to submit, view, and update work orders.

With the proposed methodology, the user can access the information using World Wide Web browsers which means that a most users with a computer will not have compatibility issues with the software. The resident will already be provided with a user name and initial password which can be changed along with other profile information such as building location, contact information, email address, and class standing.

REFERENCES

Please refer to the following list of relevant documentations:

Feasibility Study

Software Requirement Specification

All documents can be found on the ADD's website

http://matrix.csis.pace.edu:18387/pace/classpage/index.htm

Software Design Document Template

Computer Science, Berkeley College

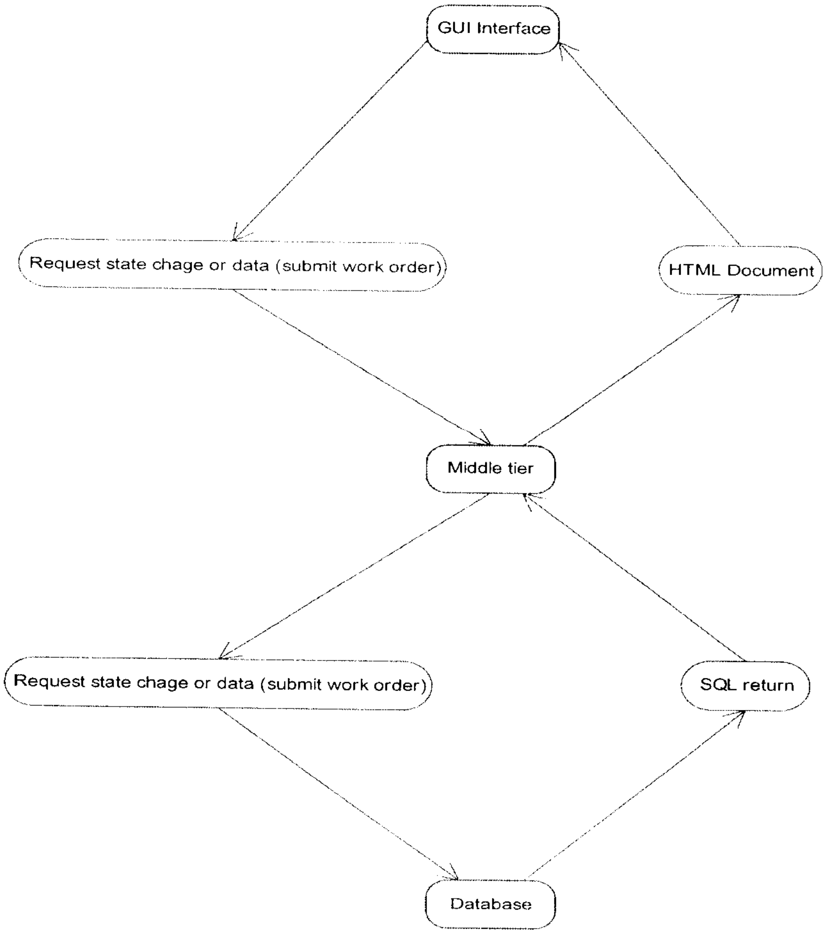
http://www-inst.eecs.berkeley.edu/ ~ cs169/downloads/templates/design-spec-v3.doc

LIMITATIONS

The resident will not be able to delete and/or modify any existing submitted work orders. The resident will not be able to change his or her user name. Only residents with valid Pace University email addresses will be allowed access to the system. Work order completion status depends on University department. Residents will not be able to access the system during off-hours. CHO Staff will not be able to add or modify users to the system (only DOH has this ability). The initial release of the system will not allow resident to send feedback after a work order has been completed.

SYSTEM METAPHOR

The overall structure of the system closely follows the Object Oriented: Call and Return architecture type. The system will be three tiered, each tier only interacting with the module directly above or below it. On the top level we have the GUI, below that is the middle tier, and at the lowest level we have the database. The GUI handles all user interaction and will be primarily an event driven system. The Database will be built as a shared repository, where the data exists in one location, and can be accessed or modified with commands from the middle tier. The GUI and the Database are held together by the middle tier, whose call and return structure brings the whole architecture together and gives it its shape. It completely abstracts all knowledge of the Database from the GUI, and all knowledge of the GUI from the Database. The Database only knows that when it wants data, it sends a request to the middle tier and receives the required document for display. The Database does not know that the GUI needs to display HTML documents; it merely returns the result from a SQL query and knows nothing of formatting requirements.



GUI Interface
Request state chage or data (submit work order)
HTML Document
Middle tier
Request state chage or data (submit work order)
SQL return
Database

Figure 1. This illustrates the overall architecture of the software. UML diagram was created with MS Visio 2002.

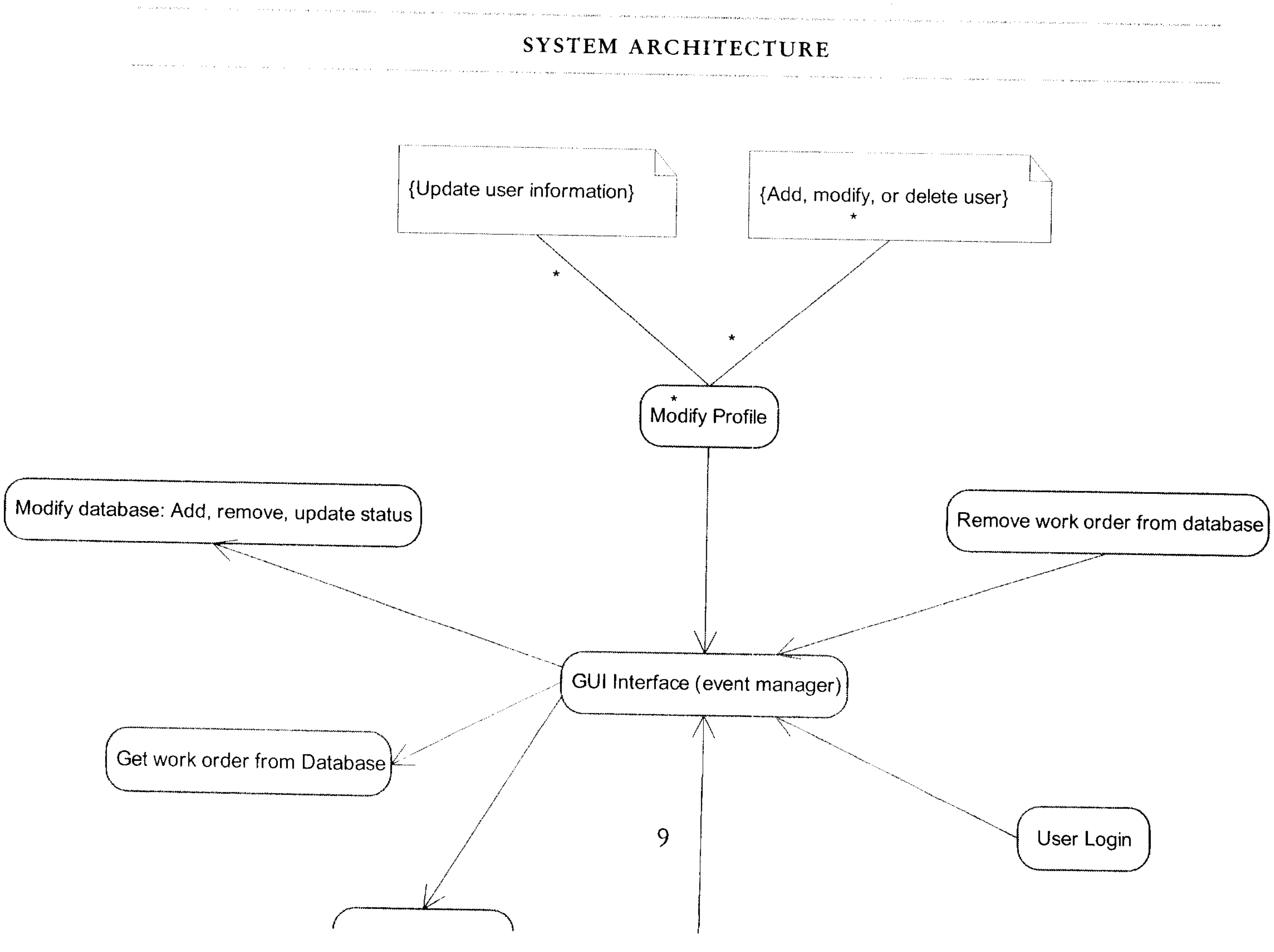

SYSTEM ARCHITECTURE
{Update user information}
{Add, modify, or delete user}
*
*
*
*
Modify Profile
Modify database: Add, remove, update status
Remove work order from database
GUI Interface (event manager)
Get work order from Database
User Login

Figure 2. This figure shows basic structure of the GUI interface. It illustrates the event driven architecture behind the GUI. UML diagram was created with MS Visio 2002.

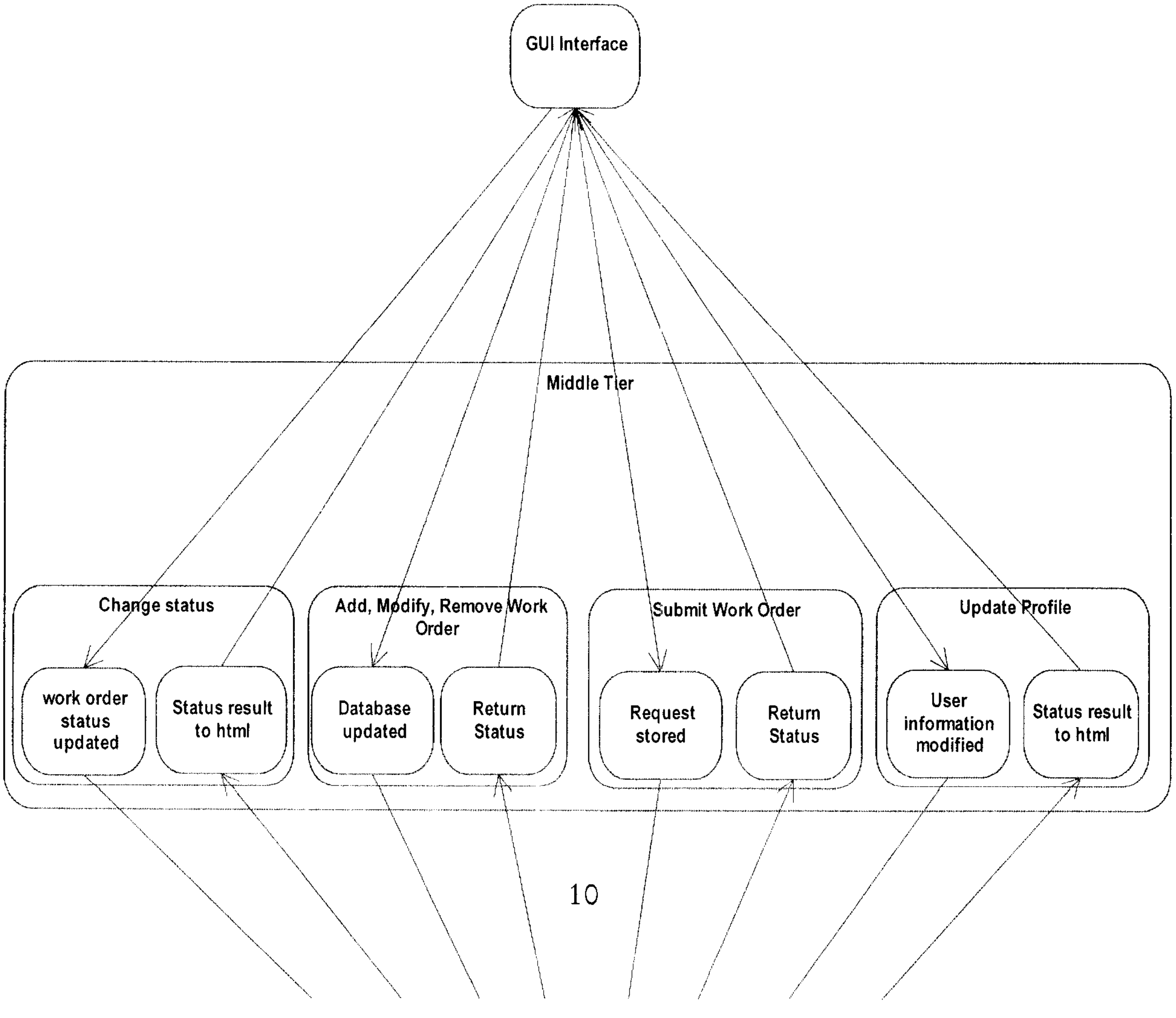


Figure 3. This shows the call and return structure of the middle tier that determines the over all system architecture. Within the middle tier structure you can see both the input action and output action for each command. UML diagram was created with MS Visio 2002.

INVARIANTS

GUI Invariants:

- Windows always remain accessible. At no point will a window that should be displayed on the screen be resized to the point they can no longer be accessed.
- Confirmation of submit request, profile modification, add, remove, or change work order status will be displayed.

Middle Tier Invariants:

- Must respond to all requests, and pass them along appropriately to the database.
- Will always return the proper HTML file to the GUI, and will give a properly formatted document for the work order display and submission.

Database Invariants:

- Maintain the database from one use to another.
- Database must never be corrupted. Must always contain proper data
- If a user request is not in the database return appropriate error.

FEATURES

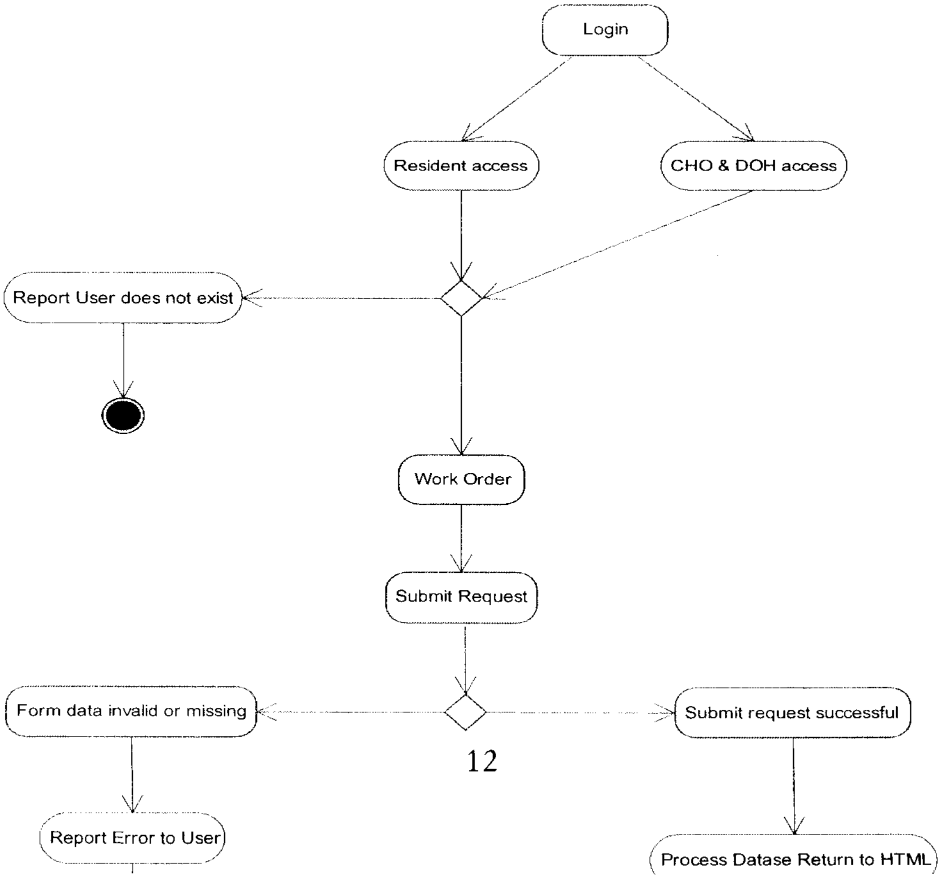

Login
Resident access
CHO & DOH access
Report User does not exist
Work Order
Submit Request
Form data invalid or missing
Submit request successful
Report Error to User
Process Datase Return to HTML

Figure 4. The Login and Work Order Submit process. UML diagram was created with MS Visio 2002.

1.1 Features - Compatibility

The greatest compatibility concerns stem from the sharing of the Internet Explorer component used for web browsing. Further compatibility issues arise when one considers that HTML is a protocol that is an ever-changing and ever-growing standard. For this reason, updates to web browsers are common, and given that we've implemented a browser control, our program will sometimes be used on a computer with an older HTML parsing engine. This could result in failures to properly load and display documents. The user also might have to download security updates and such for the IE component on their computer.

Another compatibility issue of building the Online Work Order Request System on top of a browser is that the web documents often contain components that already have behaviors of their own, especially those documents that include scripts. Clicks on a hyperlink might mean navigation is intended and they might not. Even with careful planning and design, there is still potential to confuse a user who might be expecting the wrong behavior from an event.

With the benefit of some user-friendly FAQ retrievable from the menu toolbar, the least proficient home computer users will be able to use the system.

1.2 Features - User Interface

The Online Work Order Request System will have one main window for user interaction.

Main features will be the following:

1. User login (CHO Staff, DOH, and resident access types are different).
2. Resident access:
    - Submit work order
    - Update information within profile
    - Check status of work order
3. CHO Staff access:
    - Add new work order
    - Change status of work order
4. DOH access:
    - Access includes all feature a resident and CHO Staff has
    - Add new CHO Staff Profile (grant access)
    - Remove CHO Staff Profile (revoke access)

Login:

Residents will be allowed to login with there user name and password when they access the site. CHO Staff and the DOH will be able to click on a separate link to login.

Profile view:

Residents will be allowed to update their profile and change passwords by clicking a link to another HTML page. When the resident is finished modifying profile information, they have the option to "save" profile.

Work order submit form:

The work order form will resemble the paper-based form. The resident's profile information will already be filled out on the form. The resident would be able to type in the request and click send for submission.

Work orders:

Upon login, CHO and DOH managers will be able to view a sorted list (according to ID number) of work order. This list can be sorted by resident user, date, or status.

SYSTEM ALGORITHMS

<table>
<tr><th>Algorithm Name</th><th>A1 - ResidentCreateProfile</th></tr>
<tr><td colspan="2">Before a resident can submit a work order online, he/she must be registered with the system. In the registration stage, the resident must provide all the necessary information, such as First Name, Last Name, Pace issued email Address, a password, residential hall, etc. Once the resident completes the registration form, he/she then submits the form to the system.
The system will then process the data which the resident has entered. The processing is done is</td></tr>
</table>

several stages:

1. Before the resident submits the HTML form with his/her data, JavaScript is used to validate the data. The JavaScript will be checked that the all required information is provided. The following information will be collected: All data listed, unless otherwise noted, is required for registering.
    a. First Name
    b. Last Name
    c. Pace ID Number (SS#)
    d. Pace issued email address (the User ID)
    e. A password
    f. Residential Hall
    g. Residential Hall room number
    h. Room telephone number
    i. Cellular telephone number (optional)

2. Once the JavaScript validates all the input, the data is submitted to the system for processing.

3. To insure that a resident does not try to register twice, the system will check if the resident already exists in the system. The system will check is the resident's email address already exist in the system
    a. A connection to the database is established
    b. An SQL Statement object is created.
    c. An SQL query is formatted with the resident email address
    d. The SQL query is executed against the database
    e. If the system finds that the resident's email address already exists in the database, a message will be provided stating that a profile already exists.
    f. If the system does not find the email address, a new resident profile will be created.

4. Adding the new profile to the system will be handled by a Java Servlet, *ResidentCreateProfile*.
    a. The Servlet will collect all the data items and store them in the CreateResProfile class's instance variables. Once the data is collected, the next step is accessing the database and saving the data.
    b. A connection to the database is established
    c. An SQL Statement object is created.
    d. An SQL query is formatted with the resident's profile information and the query is executed. The query will be executed via a JDBC Transaction
        i. If the SQL query does not execute successfully for any reason, then the database will not be updated – the state of the database will not change. The system will call the rollback method to preserve a stable state of the database
        ii. If the query is executed successfully, then the data is saved in the system permanently. The system will call the commit method to make the new data permanent in the system

5. A profile is created in system. The system will provide the user with a confirmation that he/she now has access to the online system. A message will also be provided stating the user should save to save his/her user id (his/her Pace email address) and his/her password and proceed to the system's login page

Algorithm Name	A2- ResidentLogin

The online system is restricted on only authorized users. Before a resident can access the system and all it's features, he/she must login with his/her User ID and Password, which they submitted when creating their profile (see algorithm #1).

The following is the procedure for a resident login into the system

Reference Algorithms: A1

1. The resident visits the Resident's Login page of the system

2. The resident enter his/her User ID and password and then proceeds to login
    a. JavaScript is used ensure the resident enters his/her User ID and password. The resident will not be allowed to proceeded if he/she does not enter both his/her User ID and password
3. The resident submits his/her User ID and password to the system for verification

4. The system will then process in the incoming data from the resident. Login to the system will be handled a Servlet, *ResidentLogin*. The Servlet will process the login as follows:
    a. The resident's User ID and password is collected from the resident and saved in the Servlet class instance variables.
    b. A global connection to the database is established
    c. An SQL Statement object is created.
    d. An SQL query is formatted with the resident's User ID and password
    e. The SQL query is executed against the database to check if the submitted User ID and Password is correct.

5. Once the SQL query is executed against the database, the database will return the results to the Servlet. The SQL query will return a result which can be evaluated in 2 possible ways:
    a. *The resident's User ID was not found in system.* This means that the resident does not have a profile on the system. The message will be provided to the resident that he/she does not have a profile and should register. This output also means that the resident might have entered this User ID incorrectly. The system will provide a message stating the submitted resident's User ID was not found in the system
    b. The resident's password is invalid. In this scenario, the resident's User ID is valid but the password is invalid. The system will notify the resident that his/her password does not match the password entered when creating his/her profile.

6. If the resident's User ID and password is verified
    a. A Session object is created to store the resident's User ID

7. The system will then forward the resident to the appropriate web page.

Algorithm Name	A3-ResidentUpdateProfile

A main feature of OLWOR System is that a resident should be able to update his/her profile at any given time. This operation is necessary since it's possible that a resident may change his/her living location and personal information during the school year. The resident will also be able to update certain information in his/her profile. The resident will also be allowed to update:

a. Password
b. Residential Hall
c. Residential Hall room number
d. Room telephone number
e. Cellular telephone number (optional)

The following is the procedure a resident updating his/her profile

Reference Algorithms: A1, A2

1. Resident logs into the system with his/her User ID and password

2. Resident Clicks on a link to update profile

3. The resident's exiting profile is read from the database and populates HTML form.
    a. Populating the HTML form with data from a database is handled with a JSP page.
    b. A global connection is made to the system's database
    c. An SQL Statement object is created
    d. An SQL query is created and formatted with the User ID taken from the Session object (created when the resident logged in).
    e. The SQL query is executed against the database.
    f. The data is then read into the HTML form.

4. The resident then updates the his/her profile and submits the HTML form to the database
    a. JavaScript is used to validate the data entered the resident
    b. The resident is required to supplied all the required information

5. Updating the resident's profile in the system's database is handled by a Servlet, *ResidentUpdateProfile*. The resident's profile will be updated as follow:
    a. Data from the HTML form with be collected and saved in the Servlet's class instance variables
    b. A SQL Statement object is created
    c. An SQL query is create and formatted with the resident's information collected to update Resident's Table in the database

d. An SQL query is executed. The query will be executed via a JDBC Transaction
    i. If the SQL query does not execute successfully for any reason, then the database will not be updated – the state of the database will not change. The system will call the rollback method to preserve a stable state of the database
    ii. If the query is executed successfully, then the data is saved in the system permanently. The system will call the commit method to make the change permanent in the system

6. A message is provided to the resident stating his/her profile has been updated.

Algorithm Name	A4-ResidentSubmitWorkRequest

This is a key operation of the online system. Once a Resident is logged in, he/she can submit a work order request via the system. The resident is required to complete a simple HTML form with a description of the work request and then submit the request to the new system. Once the request is submitted to the system,

Reference Algorithms: A1, A2

The following is the procedure for a resident to submit a work order request

1. Resident logs in to the system

2. Resident clicks on a link to submit a new work order request

3. Resident completes a simple HTML form with the required fields
    a. Resident's First & Last Name
    b. Residential Hall and Room #
    c. Email address
    d. Room Telephone number
    e. Cellular telephone number
    f. Work Request Type
        i. DoIT
        ii. B&G
        iii. S&S
        iv. CoinMac
    g. A full description of the work to be done.

4. The resident then submits the HTML form to the system for processing
    a. JavaScript is used to validate the resident's data
    b. The system will not allow the resident to submit an incomplete request. All the required data must be provided

5. The work order request is submitted to the system and saved

6. Saving the work order request will be handled by a Java Servlet, *ResidentWorkRequest.* The procedure for saving the work order request is follows
    - a. The Servlet collects all the data from the HTML form and save it in Servlet's class instance variables
    - b. A connection to the database is made
    - c. A SQL Statement object is created
    - d. An SQL query is create and formatted with the work order request information
    - e. An SQL query is executed to save the new work order request in the system. The query is executed via a JDBC Transaction
        - i. If the SQL query does not execute successfully for any reason, then the database will not be updated – the state of the database will not change. The system will call the rollback method to preserve a stable state of the database. The work order request will not be added to the system. The resident will be provided with a message of the error.
        - ii. If the query is executed successfully, then the new request is saved in the system permanently. The system will call the commit method to make the change permanent in the system.
        - iii. Once the request is saved in the system's database, the system will return a unique *Work Order Request ID* (WORID) that so that the student can check the status of the work order at a later time
    - f. Once the request has been saved in the database, the system will compile an email message to send to the resident and CHO Staff notifying them about the new work order
    - g. The email message will contain the following information:
        - i. The entire work order request, as submitted by the resident
        - ii. The resident's personal information
    - h. Since personal information about the resident is already saved in the system, the system did not ask the resident to submit his information why submitting his/her work order request. The system will generate the email message using data which was submitted and data from the database
    - i. Collecting the Resident's information
        - i. An SQL query is created and formatted with the Resident's User ID (captured from the Session object) to query the database
        - ii. The SQL query is executed and the resident's data is saved for the email message
    - j. Sending email
        - i. Once all the necessary data has be collected, compile the email message and send email to resident and CHO Staff member

A new work order request has been successfully submitted

Algorithm Name	A5-ResidentCheckRequestStatus
This is another key feature of the system. Once a work order request has been submitted, a	

resident can check the status of the work order's status. To do so, the resident will need the *Work Order Request ID* given after the work order was submitted. This is the only information required to check the status of a work order. The work order will have 3 possible statuses:

1. In Progress
2. Completed
3. Rejected

The status of the work order will be updated with the CHO Staff Members.

The algorithm for checking the status of a submitted work order request is as follows:
Reference Algorithms: A1, A2, A4

1. Resident logs in to the system

2. Resident requests the page for checking the status of the work order request

3. Resident enters the *WORID* given after he/she submitted the work order request

4. Checking the status of the request will be handled by a JSP page, *ResidentCheck RequestStatus.*
    - *a.* The JSP page will capture the WORID which the resident has entered
    - *b.* A connection to the database is made
    - *c.* An SQL query is created and formatted with the WORID to query the Work Order Request table in the database.
    - *d.* The result is saved in a Boolean variable
    - *e.* If the Boolean variable is true:
        - *i.* The request was found in the database
        - *ii.* The entire request will be printed on the screen for the resident's view, along with the status of the request
    - *f.* If the Boolean variable is false:
        - *i.* The request was not found in the system. This means either the request was never filed or the resident entered an incorrect WORID
        - *ii.* A message will be printed to the resident stating request not found.

Table Name	**Residents**
SQL Code	

```
CREATE TABLE Residents (
    RES_ID      VARCHAR(50) NOT NULL,
     Password        VARCHAR(20) NULL,
     First_Name        VARCHAR(50) NOT NULL,
     Last_Name         VARCHAR(50) NOT NULL,
     Res_Hall_ID       INTEGER NOT NULL,
     Pace_Stu_ID       INTEGER NOT NULL,
     Room_Number         VARCHAR(20) NULL,
     Tele_Extension     VARCHAR(20) NULL,
     Celluar_Number      VARCHAR(30) NULL,
);

ALTER TABLE Residents ADD  ( PRIMARY KEY (RES_ID) ) ;

ALTER TABLE Residents  ADD  ( FOREIGN KEY (Res_Hall_ID)
                     REFERENCES ResidentHalls) ;
```

Description

NAME	**DATA TYPE**	**Comment**
RES_ID	VARCHAR, NOT NULL	The Resident's ID will be a Pace issued email address – with domain "pace.edu." The email address will also be the primary key for the Resident's table in the system since each email

		issued by page is unique to only 1 student. The email field is required.
Password	VARCHAR, NOT NULL	The password will be alphanumeric with a maximum length 20 characters and minimum length of 6
First_Name	VARCHAR, NOT NULL	Resident's first name is required by the system. This field is required
Last_Name	VARCHAR, NOT NULL	Resident's first last is required by the system. This field is required
Res_Hall_ID	INTEGER	The resident's living location, Res_Hall_ID, will be joined to ResidentHalls table. Res_Hall_ID will be the foreign key of this table
Pace_Stu_ID	INTEGER, NOT NULL	Pace_Stu_ID will be the residents Social Security number. This field is required
Room_Number	VARCHAR	Room_Number is the resident's room number in his/her respective residential hall
Tele_Extension	VARCHAR	Tele_Extension is the resident's room telephone in his/her respective residential room.
Celluar_Number	INTEGER	Cellular_Number will be the resident's cellular phone number incase his/her room telephone number is not working. This field is not required

Table Name	**ResidentHalls**

SQL Code

```
CREATE TABLE ResidentHalls (
    Res_Hall_ID         INTEGER NOT NULL AUTO_INCREMENT,
    Res_Hall_Name        varCHAR(30) Not NULL,
    Res_Hall_Location    VARCHAR(30) Not NULL,
);

ALTER TABLE ResidentHalls
    ADD  ( PRIMARY KEY (Res_Hall_ID) ) ;
```

Description

NAME	**DATA TYPE**	**Comment**
Res_Hall_ID	INTEGER, NOT NULL	When a new residential hall added to this table, a new, unique ID will be given to the record. Res_Hall_ID will be primary key for this table.
Res_Hall_Name	VARCHAR, NOT NULL	The residential hall must have a name, for example, "Maria's Tower 2." The data will be used to populate HTML form its. This field is required.
Res_Hall_Location	VARCHAR, NOT NULL	The location of the hall.

Table Name	RAProfile (Residential Advisors Profiles)
SQL Code	CREATE TABLE RAProfile (

```
    RAID              INTEGER NOT NULL AUTO_INCREMENT,
    RA_First_Name        VARCHAR(50) NOT NULL,
    RA_Last_Name         VARCHAR(50) NOT NULL,
    RA_Email          VARCHAR(30) NOT NULL,
    RA_Telephone         VARCHAR(20) NULL
);

ALTER TABLE RAProfile
    ADD  ( PRIMARY KEY (RAID) ) ;
```

Description

NAME	DATA TYPE	Comment
RAID	INTEGER, NOT NULL	When a new RA is added to this table, a new, unique ID will be given to the record. RAID will be primary key for this table.
RA_First_Name	VARCHAR, NOT NULL	First name of the RA. This field is required with a maximum of 50 characters
RA_Last_Name	VARCHAR, NOT NULL	Last name of the RA. This field is required with a maximum of 50 characters
RA_Email	VARCHAR, NOT NULL	Email address of the RA. This field is required with a maximum of 30 characters
RA_Telephone	VARCHAR, NULL	The RA's telephone. This field is not required

Table Name	RHCProfiles (Residential Hall Coordinators Profiles)

SQL Code

```
CREATE TABLE RHCProfiles (
    RCHID             INTEGER NOT NULL AUTO_INCREMENT,
    RCH_First_Name      VARCHAR2(50) NOT NULL,
    RCH_Last_Name       VARCHAR2(50) NOT NULL,
    RCH_Email        VARCHAR2(30) NOT NULL,
    RCH_Telephone       VARCHAR2(30) NULL
);

ALTER TABLE RHCProfiles
    ADD  ( PRIMARY KEY (RCHID) ) ;
```

Description

NAME	DATA TYPE	Comment
RCHID	INTEGER, NOT NULL	When a new RCH is added to this table, a new, unique ID will be given to the record. RCHID will be primary key for this table.
RCH_First_Name	VARCHAR, NOT NULL	First name of the RCH. This field is required with a maximum of 50 characters
RCH_Last_Name	VARCHAR,	Last name of the RCH. This field is required with a

	NOT NULL	maximum of 50 characters
RCH_Email	VARCHAR, NOT NULL	Email address of the RA. This field is required with a maximum of 30 characters
RCH_Telephone	VARCHAR, NULL	The RA's telephone. This field is not required

Table Name	**BuildingRAdvisors**
SQL Code	

```
/* Each building will have an RA. This table is populating by selecting a
Residential Hall and an RA from the RA Profile table*/


CREATE TABLE BuildingRAdvisors (
     Res_Hall_ID       INTEGER NOT NULL,
     RAID            INTEGER NOT NULL
);

ALTER TABLE BuildingRAdvisors
     ADD  ( PRIMARY KEY (Res_Hall_ID, RAID) ) ;

ALTER TABLE BuildingRAdvisors
     ADD  ( FOREIGN KEY (RAID)
                    REFERENCES RAProfile ) ;

ALTER TABLE BuildingRAdvisors
     ADD  ( FOREIGN KEY (Res_Hall_ID)
                    REFERENCES ResidentHalls ) ;
```

Description

NAME	DATA TYPE	Comment
Res_Hall_ID	INTEGER, NOT NULL	Res_Hall_ID is pulled from the Residential Hall table. This will be both a primary key and a foreign key
RAID	NTEGER, NOT NULL	RAID is pulled from the RAProfile table. This will be both a primary key and a foreign key

Table Name	**BuildingRHCord**
SQL Code	

```
/* Each building will have an RCH. This table is populating by selecting a
Residential Hall and an RCH from the RHCProfiles table*/


CREATE TABLE BuildingRHCord (
     Res_Hall_ID       INTEGER NOT NULL,
     RCHID            INTEGER NOT NULL
);
```

```
ALTER TABLE BuildingRHCord
     ADD  ( PRIMARY KEY (Res_Hall_ID, RCHID) ) ;

ALTER TABLE BuildingRHCord
     ADD  ( FOREIGN KEY (RCHID)
                    REFERENCES RHCProfiles ) ;


ALTER TABLE BuildingRHCord
     ADD  ( FOREIGN KEY (Res_Hall_ID)
                    REFERENCES ResidentHalls ) ;
```

Description

NAME	DATA TYPE	Comment
Res_Hall_ID	INTEGER, NOT NULL	Res_Hall_ID is pulled from the Residential Hall table. This will be both a primary key and a foreign key
RCHID	NTEGER, NOT NULL	RCHID is pulled from the RHCProfiles table. This will be both a primary key and a foreign key

Table Name	CHOStaff
SQL Code	

```
/* This table will store information for both CHO Staff and DOH. */


CREATE TABLE CHOStaff (
     CHOID            INTEGER NOT NULL AUTO_INCREMENT,
     CHO_First_Name      VARCHAR(50) NOT NULL,
     CHO_Last_Name       VARCHAR(50) NOT NULL,
     Email_Address      VARCHAR(30) NOT NULL,
     Password        VARCHAR(20) NOT NULL,
     Access_Level      INTEGER NOT NULL,
     Last_Edit_Date      DATE NULL
);

ALTER TABLE CHOStaff
     ADD  ( PRIMARY KEY (CHOID) ) ;
```

Description

NAME	DATA TYPE	Comment
CHOID	INTEGER,	When a new Staff member is added to this table, a new,

	NOT NULL	unique ID will be given to the record. CHOID will be primary key for this table.
CHO_First_Name	VARCHAR NOT NULL	Staff's first name. This field is required
CHO_Last_Name	VARCHAR NOT NULL	Staff's last name. This field is required
Email_Address	VARCHAR NOT NULL	Staff's email address. This field is required
Password	VARCHAR NOT NULL	Password will be alpha numeric with a maximum of 20 characters
Access_Level	INTEGER NOT NULL	There will be 3 levels of access. By default, residents will be level `1`, the lowest possible access. The next level will ne
Last_Edit_Date	DATE	If the Staff profile was updated, capture the date of update

Table Name	WorkRequestTypes
SQL Code	

```
/* this table with hold the different types of work request, such as B&G,
S&S, DoIT, CoinMac. If the Staff adds a new work request type, the data
will be stored here. The data in this table is also used for populating the
Work Order Submit page - The Request types are dynamically read into
the HTML form */

CREATE TABLE WorkRequestTypes (
     ReqType_ID INTEGER NOT NULL AUTO_INCREMENT,
     Work_Type_Name       VARCHAR(50) Not NULL,
     Work_Type_Description VARCHAR(1000) NULL,
     Work_Type_Contact    VARCHAR(30) NULL,
    Send_Request_to      VARCHAR(30) NULL
);

ALTER TABLE WorkRequestTypes
     ADD  ( PRIMARY KEY (Work_Request_Type_ID) ) ;
```

Description

NAME	DATA TYPE	Comment
ReqType_ID	INTEGER, NOT NULL	When a new Work Request Type is added to this table, a new, unique ID will be given to the record. ReqType_ID will be primary key for this table.
Work_Type_Name	VARCHAR NOT NULL	The name of the request. For example, DoIT
Work_Type_	VARCHAR	A description of work type. For example, DoIT work

Description	NULL	orders relate to LAN issues, Ethernet problems, etc. The maximum number of characters allowed for this field is 1000.
Work_Type_Contact	VARCHAR NULL	The contact for the work type
Send_Request_to	INTEGER NOT NULL	This field will determine which Staff member will receive work orders of this type.

Table Name	**WorkOrders**
SQL Code	

```
/* All work orders will be saved in this table. The table has several foreign
keys to other tables. */

CREATE TABLE WorkOrders (
    Work_Order_ID        VARCHAR(100) NOT NULL,
    WOR_SubmitDate           DATE NOT NULL,
    WOR_Description      VARCHAR(3000) NOT NULL,
    RES_ID         INTEGER NOT NULL,
    CHOID              INTEGER NOT NULL,
    Res_Hall_ID        INTEGER NOT NULL,
    ReqType_ID INTEGER NOT NULL,
    Submitted_by    VARCHAR(20) NOT NULL,
    Work_Order_Status   VARCHAR(35) NOT NULL,
    Last_Edit_Date       DATE NULL
);

ALTER TABLE WorkOrders
    ADD  ( PRIMARY KEY (Work_Order_ID) ) ;

ALTER TABLE WorkOrders
    ADD  ( FOREIGN KEY (ReqType_ID)
                REFERENCES WorkRequestTypes ) ;

ALTER TABLE WorkOrders
    ADD  ( FOREIGN KEY (Res_Hall_ID)
                REFERENCES ResidentHalls ) ;

ALTER TABLE WorkOrders
    ADD  ( FOREIGN KEY (CHOID)
                REFERENCES CHOStaff ) ;

ALTER TABLE RES_ID
    ADD  ( FOREIGN KEY (RES_ID)
```

	REFERENCES Residents) ;	
Description		
NAME	**DATA TYPE**	**Comment**
Work_Order_ID	VARCHAR NOT NULL	
WOR_SubmitDate	Date NOT NULL	The date the work order was submitted. This field is required
WOR_Description	VARCHAR NOT NULL	A description of work order. This field is required. A maximum of 3000 characters is accepted for this field
RES_ID	Integer NOT NULL	The Resident's id. This field is required and a foreign key in this table. All info about the resident can be captured with his/id
CHOID	Integer NULL	This field will be modified when a CHO updates the status of work order. The Staff ID will be added to reflect the change. This field is required and a foreign key in this table
Res_Hall_ID	INTEGER NOT NULL	The residential hall where the work has to be done. This field is required and a foreign key in this table
ReqType_ID	DATE	The type of work order request. This field is required and a foreign key in this table
Submitted_by	VARCHAR NOT NULL	A work order can be submitted by either a staff member or resident. This field will accept only two possible inputs (By Resident, By Staff).
Work_Order_Status	VARCHAR NOT NULL	The work order status. The status can be either 'In Progress', 'Completed' or 'Rejected'. This field is required
Last_Edit_Date		The last time a request was edited or updated

GLOSSARY

CHO Atff = Campus Housing Operations Staff members

DOH = Director of Housing.

GUI = Graphical User Interface.

Front/First Tier = The user interface.

Middle Tier = Handles communication between the GUI interface and Database.

Third Tier = The System's Database

Invariant = An assertion that must be true both before and after the execution of each operation.

Profile = Information about each residents, and will contain full name, residential hall, room number, contact information, ID number, and class standing .