Occupancy Engine

System Manual

Team Project Group 8

April 24, 2015

Revision Sheet

Release No.	Date	Revision Description
1.0	4/24/2015	Initial Revision

SYSTEM MANUAL

TABLE OF CONTENTS

1. GENERAL INFORMATION

1.1. System Overview

The Occupancy Engine system is intended to accurately quantify the number of people in each room of a selected building. It is based on the information coming from different sensors. All the received data is stored into database. This historical data is used for different queries and analyses. The system provides user interface to show the current people presence in every room. It also provides different reports and executes queries according to the data search, entered by the user. Based on the historical data, the system performs further analyses for the calculation of the occupancy pattern and for the future occupancy prediction. The system also calculates confidence level for a number of people in a room by using all the confidence levels for each of the doors, connected to this room. First for the confidence level of the doors is used confidence level, coming from the sensors. When enough data is gathered, the confidence level will be calculated, based on the number of people, moved through the particular room and the automatic corrections, made from the system for the doors, connected to the same room. The system makes automatic correction of the errors. Error handling module performs automatic check-errors and correct-errors functions. Automatic procedures for checking errors and making errors correction are executed after every transmission of data. There is also checking rules procedure, which is scheduled periodically at a pre-defined time and interval (i.e. every night at 00.00) and performs automatic checking and error corrections according to user-defined rules. Using these rules, the system will be able to determine wrong activities in a room and correct them, tracing back to the entrance/exit of the building using breath first search algorithm. Later on, when there is enough data for analyses a system could creates its own set of rules based on the historical data. All automatic corrections, made by the system, are stored in the database for further analyses. Set of tables are provided for the system parameters for rules, crowdedness, period and time for scheduled procedures for error handling module and occupancy convex calculation.

The software delivered in this release is fully functional to receive and store data from different type of sensors, to quantify number of people and calculate confidence level of people presence in a room and building, to calculate convex for an occupancy pattern, error handling module and set of different reports. New algorithms and features can be developed by Team 8 with the next releases.

1.2. Points of Contact

For additional information, Team Group 8 can be contacted through Project Leader Aleksandar Rusinov (aleksandar.rusinov.13@ucl.ac.uk).

2. SYSTEM SUMMARY

2.1. System Configuration

The system is web based, it contains modules for data receiving, storing and processing, MySQL database, web site with responsive design built using foundation framework. Additional modules for system initialization and simulator of data transmission from hardware sensors are provided. Overall structure of the system is shown in Figure 1.1

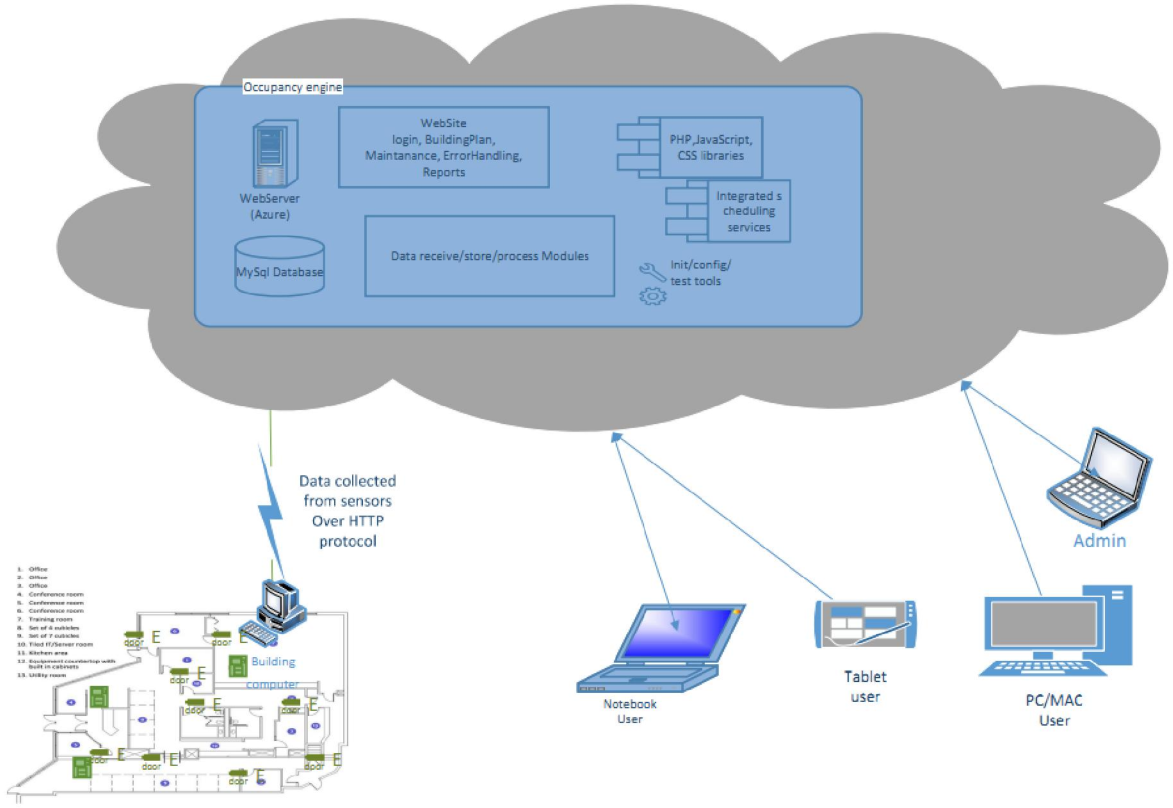

Figure 2-1 System overview schema

1. Our product is a cloud based system:
    - Server side – Microsoft Azure cloud platform
    - MySQL database server
    - PHP scheduled modules to receive, store and process input data.
    - Scheduled procedures for errors handling and convex calculation
    - HTTP transfer to collect data send from an Arduino controlled sensors
2. User interface using foundation framework responsive design, HTML5, CSS, JavaScript
3. Modules for:
    - Initializing Java module - for set up, database creation, configuration and initial data loading in system parameter tables.
    - Java Client side module - Simulator of sending data online from hardware sensors
    - PHP module to create and load test samples of historical data
4. Github repository is used for version and source control from where the source code is deployed on Azure and local host staging environment for the product and its iterations.

3. GETTING STARTED

3.1. System prerequisites

Extracting, Installing, and Running Occupancy Engine

Prerequisites – server environment: web server (Apache, Azure, etc.), MySQL database server, PHP interpreter on server side, access rights to: deploy system site in public document accessible folder on server, administrative credential rights to create and setup MySQL database, configure and schedule jobs i.e. CRON/WEB Jobs/AT

Occupancy engine system is web based and on client side customer needs standard browser and internet access. User will be provided with an access credentials – username and password. System is compatible with the latest versions of the Google Chrome, Firefox, Internet Explorer, Opera, and Safari browsers. User interface is based on open-source framework platform responsive by design - that means it can be used on variety of hardware and software platforms - from large desktop screens to mobile devices.

3.2. System installation

Deploy system from GitHub repository.

Set credentials in *dbconnect.php* and *dbconnectPDO.php* for the database, server URL and user and password.

Fill JSON script with description of building plan.

Run Java program to apply JSON file for populating database tables, procedures and to set initial configuration data – building plan and parameters.

3.3. Setup parameters and initializing

Start browser, fill system URL in address bar and login as admin. Hit Mntnances link on left side navigation bar.
Fill in Forcedrules, Convex definition, roomparameters, occupancylevels concrete values to reflect building plan and expected occupants presence.
Sent to admin of sensors system URLs of data gathering php modules and building plan configuration for room/door IDs.

4. DATABASE STRUCTURE

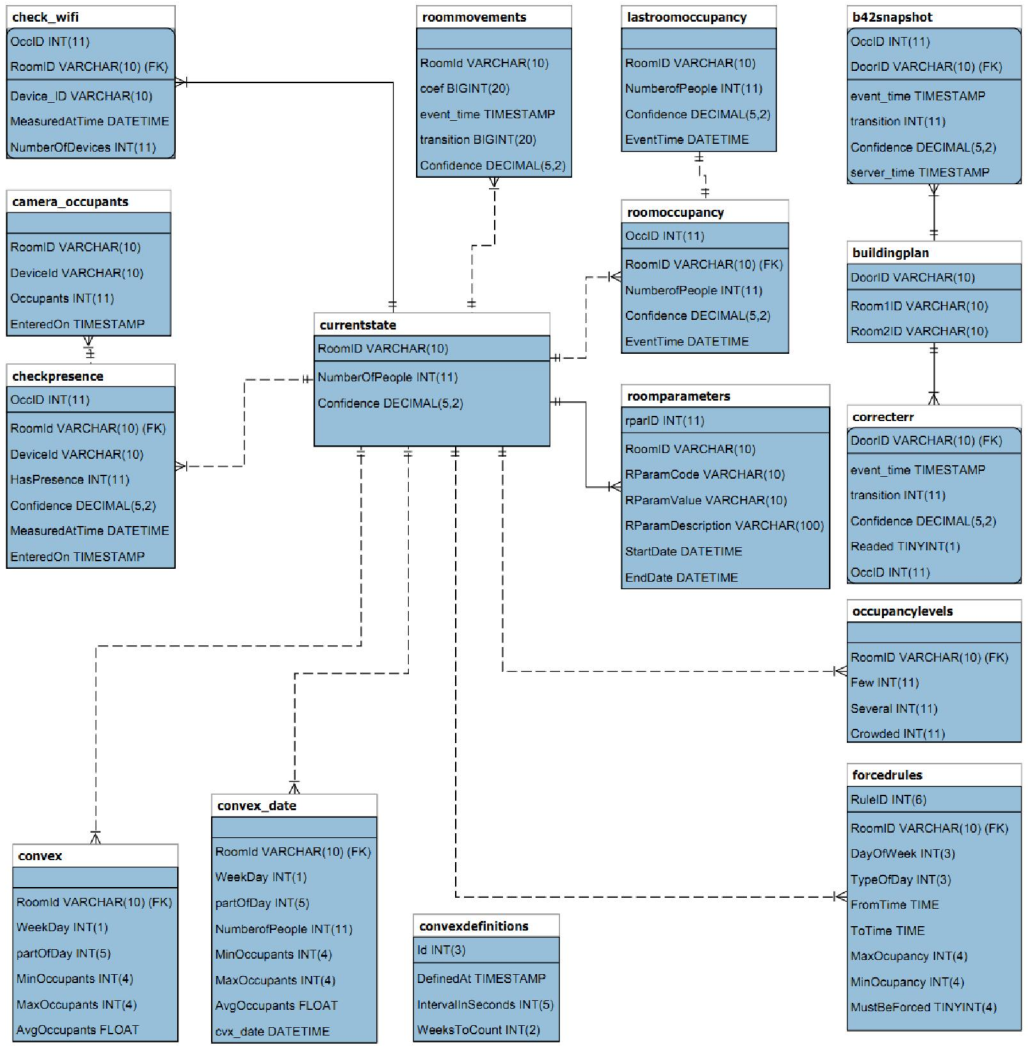

Figure 4-1 Database entity-relationship diagram

B42Snapshot stores the data that comes directly from the Arduino. It shows the transition at a given door along with a confidence, direction sign and timestamp of the event.

BuildingPlan lists all the doors in a building as well as their adjacent rooms. For this model we assume that a door is two sided. The transition is positive when it is from Room1 to Room2 and negative if it is from opposite direction.

CurrentState stores the number of people currently occupied a given room, along with a cumulative confidence level. This table is created to allow faster access to the current state of each room.

RoomOccupancy stores the past data for each room occupancy at a given moment. A parameter time interval is used, for example 1 day, 1 hour or 5 minutes depending on the people movement dynamic and the snapshot will be generated. This database will be updated during a period where there are no activities. For example, very late at night.

OccupancyLevels gives information about the crowdedness of each room. The values vary depending on the area of a room. The occupancy level is associated with a color. The values in Empty, Few, Several, Crowded present an upper bound for the level. Empty is always 0. For example if we have Few – 4, Several – 10, Crowded – 30, and we have green for Empty, yellow for Few, orange for several and red for Crowded and the system shows that there are 14 people in a room, it will have red label.

RoomParameters is a system table that specifies various characteristics about the rooms. For example, one can store HVAC (heating regulation and air-conditioning) conditions about every room. Since that is a major energy consumer for buildings a demand driven control will be of essential value. The value parameter represents the number of people that should be in the room for a certain activity to start. The dates show the period that this activity is valid.

CheckPresence stores the data that comes directly from the room sensors. It shows the presence (Yes/No) in a given room along with a device number, confidence and timestamp of measurement.
Check_WiFi stores the data that comes directly from the Wi-Fi sensors. It shows the number of devices in a given room along with a device number and timestamp of measurement.

CorrectErr stores a correction transitions of error handling module. Structure and fields are same as of b42snapshot.

ForcedRules is a table that specifies max and min people limits for the room at agiven period of a given day of week, typeofday (working or holiday), and flag for forcing rule. This is used by scheduled error correction procedure to enforce sets limit.

ConvexDefinitions stores the parameters used in building and populating convex and convex_date tables. This parameters are length in weeks in the past and interval length to divide days (in seconds).

Convex_date stores calculated history of rooms occupancy per defined weeks in the past in form of min, max, avg values per intervals of a days.

Convex stores min, max and avg values of rooms occupancy calculated per intervals of a weekday.
There are several helper views for facilitating queries and data processing procedures:
roommovements, **camera_occupants** and **lastroomoccupancy**.

b42snapshot – store transition records from door sensors

Column	Type	Null	Default	Comments	MIME
OccID	int(11)	No		ID	
DoorID	varchar(10)	Yes	NULL	Door ID	
event_time	timestamp	Yes	NULL	Event time	
transition	int(11)	Yes	NULL	Transition with sign	
Confidence	decimal(5,2)	Yes	NULL	Confidence	
server_time	timestamp	Yes	NULL	Server time	

Buildingplan – describe doors and rooms configuration

Column	Type	Null	Default	Comments	MIME
DoorID	varchar(15)	No		Door ID	
Room1ID	varchar(10)	Yes	NULL	Room ID – room 'out' for this door	
Room2ID	varchar(10)	Yes	NULL	Room ID – room 'in' for this door	

camera_occupants – VIEW for last record from room devices per room

Table comments: VIEW

Column	Type	Null	Default	Comments	MIME
RoomID	varchar(10)	Yes	NULL	Room ID	
DeviceId	varchar(10)	Yes	NULL	Device ID	
Occupants	int(11)	Yes	NULL	Number of occupants	
EnteredOn	timestamp	No	0000-00-00 00:00:00	Event time	

check wifi - store records from WiFi sensors

Column	Type	Null	Default	Comments	MIME
OccID	int(11)	No		ID	

RoomID	varchar(10)	Yes	NULL	Room ID	
Device ID	varchar(10)	Yes	NULL	WI-FI Device ID	
MeasuredAtT ime	datetime	Yes	NULL	Event time	
NumberOfDevices	int(11)	Yes	NULL	Number of occupants	

Checkpresence - store presence records from room sensors

Column	Type	Null	Default	Comments	MIME
OccID	int(11)	No		ID	
RoomId	varchar(10)	Yes	NULL	Room ID	
DeviceId	varchar(10)	Yes	NULL	Device ID	
HasPresence	int(11)	Yes	NULL	Flag for presence	
Confidence	decimal(5,2)	Yes	NULL	Confidence	
MeasuredAtT ime	datetime	Yes	NULL	Event time	
EnteredOn	timestamp	No	CURRENT_TIMESTAMP	Entered time	

Convex – generated avg occupancy and min/Max convex for past period and on intervals, defined in convexdefinitions

Column	Type	Null	Default	Comments	MIME
RoomId	varchar(10)	No		Room ID	
WeekDay	int(1)	No		Day of week	
partOfDay	int(5)	No		Time interval of the day	
MinOccupants	int(4)	No		Min value for this interval	
MaxOccupants	int(4)	No		Max value	
AvgOccupants	float	No		Average value	

convex data – history data for convex

Column	Type	Null	Default	Comments	MIME
RoomId	varchar(10)	No		Room ID	
WeekDay	int(1)	No		Day of week	
partOfDay	int(5)	No		Time interval of the day	
NumberofPeople	int(11)	No		Number of people	
MinOccupants	int(4)	No		Min occupants	
MaxOccupants	int(4)	No		Max occupants	
AvgOccupants	float	No		Average occupants	
cvx_date	datetime	No		Convex date	

convexdefinitions

To keep weeks and time intervals for convex calc

Column	Type	Null	Default	Comments	MIME
Id	int(3)	No		ID	
DefinedAt	timestamp	No	CURRENT_TIMEST AMP	Current time	
IntervalInSeconds	int(5)	No	900	Time interval in seconds	
WeeksToCount	int(2)	No	10	Number of weeks to count	

Correcterr – store records from error correction procedures

Column	Type	Null	Default	Comments	MIME
DoorID	varchar(15)	Yes	NULL	Door ID	
event_time	timestamp	No	CURRENT_TIMESTAMP	Event time	
transition	int(11)	Yes	NULL	Transition with sign	
Confidence	decimal(5,2)	Yes	NULL	Confidence	
Readed	tinyint(1)	Yes	NULL	Flag for read	
OccID	int(11)	No		ID	

Currentstate – utility table with current presence in rooms

Column	Type	Null	Default	Comments	MIME
RoomID	varchar(10)	No		Room ID	
NumberOfPeople	int(11)	No		Number of people	
Confidence	decimal(5,2)	No		Confidence	

Forcedrules – rules for min/max occupancy per interval of weekday to be used in scheduled procedure for night error correction

Column	Type	Null	Default	Comments	MIME
RuleID	int(6)	No		Rule ID	
RoomID	varchar(10)	No		Room ID	
DayOfW eek	int(3)	No		Day of week	
TypeOfDay	int(3)	No		Type of day (workday or not)	
FromTime	time	No		Start of time interval	
ToTime	time	No		End of time interval	
MaxOcupancy	int(4)	No		Rule for Max occupancy	
MinOcupancy	int(4)	No		Rule for Min occupancy	
MustBeForced	tinyint(4)	No	0	Rule for Average occupancy	

Lastroomoccupancy – VIEW – last record in roomoccupancy per room

Column	Type	Null	Default	Comments	MIME
RoomID	varchar(10)	Yes	NULL	Room ID	
NumberofPeople	int(11)	Yes	NULL	Number of people	
Confidence	decimal(5,2)	Yes	NULL	Confidence	
EventTime	datetime	Yes	NULL	Event time	

Occupancylevels – predefined levels of room presence (crowdedness)

Column	Type	Null	Default	Comments	MIME
RoomID	varchar(10)	No		Room ID	
Few	int(11)	No		'Few' value for crowdedness	
Several	int(11)	No		'Several' value for crowdedness	
Crowded	int(11)	No		'Crowded' value for crowdedness	

Roommovements – VIEW combining data from b42snapshot and Correcterr for transition events per room

Table comments: VIEW

Column	Type	Null	Default	Comments	MIME
RoomId	varchar(10)	Yes	NULL	Room ID	
coef	bigint(20)	No	0	Coefficient	
event_time	timestamp	Yes	NULL	Event time	
transition	bigint(20)	Yes	NULL	Transition with sign	
Confidence	decimal(5,2)	Yes	NULL	Confidence	

Roomoccupancy – utility table – periodically saved snapshot of currentstate with calculated confidence field

Column	Type	Null	Default	Comments	MIME
OccID	int(11)	No		ID	
RoomID	varchar(10)	Yes	NULL	Room ID	
NumberofPeople	int(11)	Yes	NULL	Number of people	
Confidence	decimal(5,2)	Yes	NULL	Confidence	
EventTime	datetime	Yes	NULL	Event time	

Roomparameters – various parameters of type “key”->”value” constrained per datetime period

Column	Type	Null	Default	Comments	MIME
rparID	int(11)	No		Room Parameter ID	
RoomID	varchar(10)	No		Room ID	
RParamCode	varchar(10)	No		Room Parameter code	
RParamValue	varchar(10)	No		Room Parameter value	
RParamDescription	varchar(100)	No		Room Parameter description	
StartDate	datetime	No		Start date for this parameter value	
EndDate	datetime	No		Start date for this parameter value	

5. DATA SENDING MECHANISM

The below diagrams describe flow of the process of sending sensor data form monitored building to server with installed QOE. System supports data from 3 types of sensors – Doors, Room and Wi-Fi.

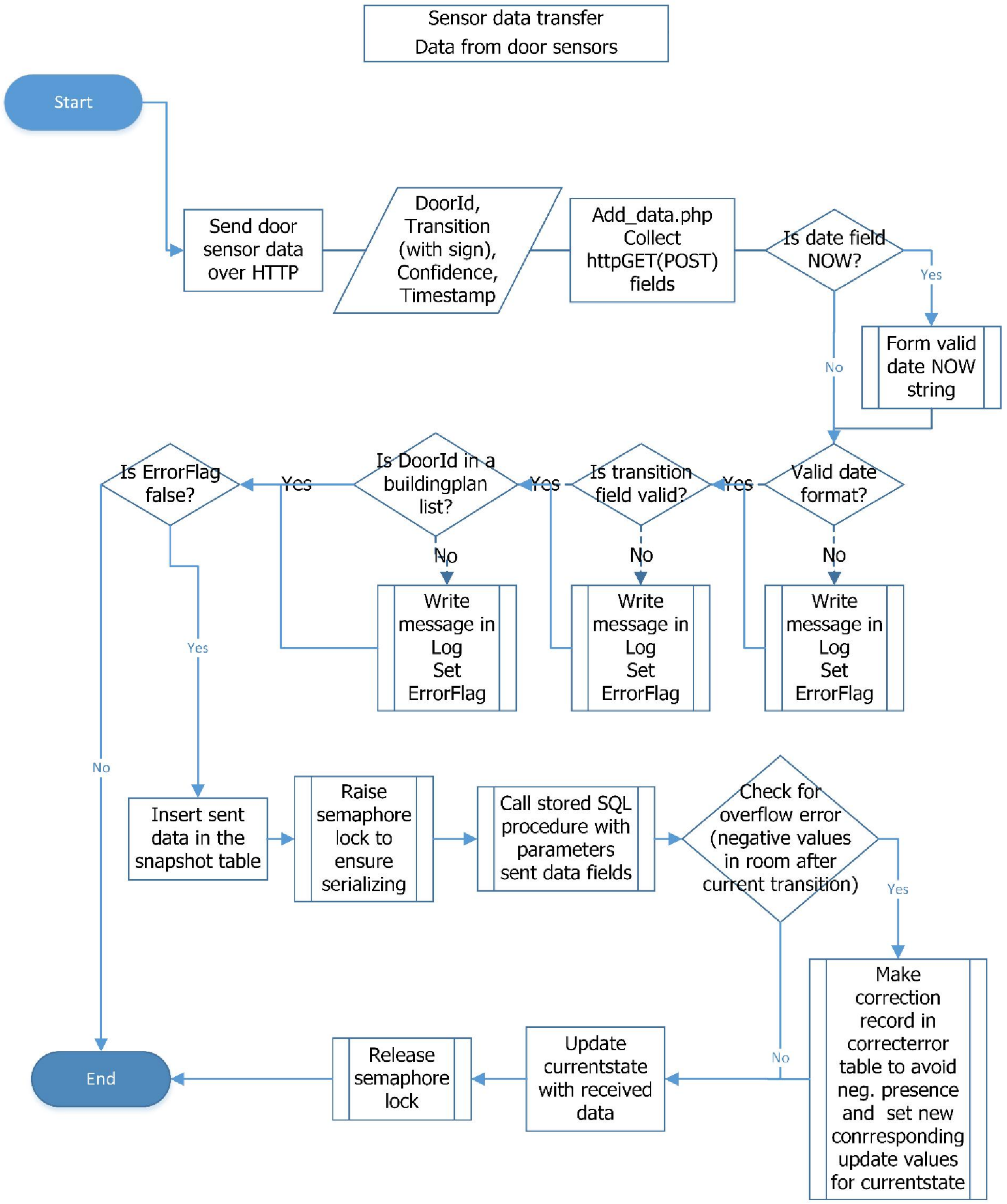
Sensor data transfer
Data from door sensors
Start
Send door sensor data over HTTP
DoorId, Transition (with sign), Confidence, Timestamp
Add_data.php Collect httpGET(POST) fields
Is date field NOW?
Yes
Form valid date NOW string
No
Valid date format?
Yes
No
Write message in Log Set ErrorFlag
Is transition field valid?
Yes
No
Write message in Log Set ErrorFlag
Is DoorId in a buildingplan list?
Yes
No
Write message in Log Set ErrorFlag
Is ErrorFlag false?
Yes
No
Insert sent data in the snapshot table
Raise semaphore lock to ensure serializing
Call stored SQL procedure with parameters sent data fields
Check for overflow error (negative values in room after current transition)
Yes
No
Make correction record in correcterror table to avoid neg. presence and set new conrresponding update values for currentstate
Update currentstate with received data
Release semaphore lock
End

Figure 5-1 Send data from door sensors

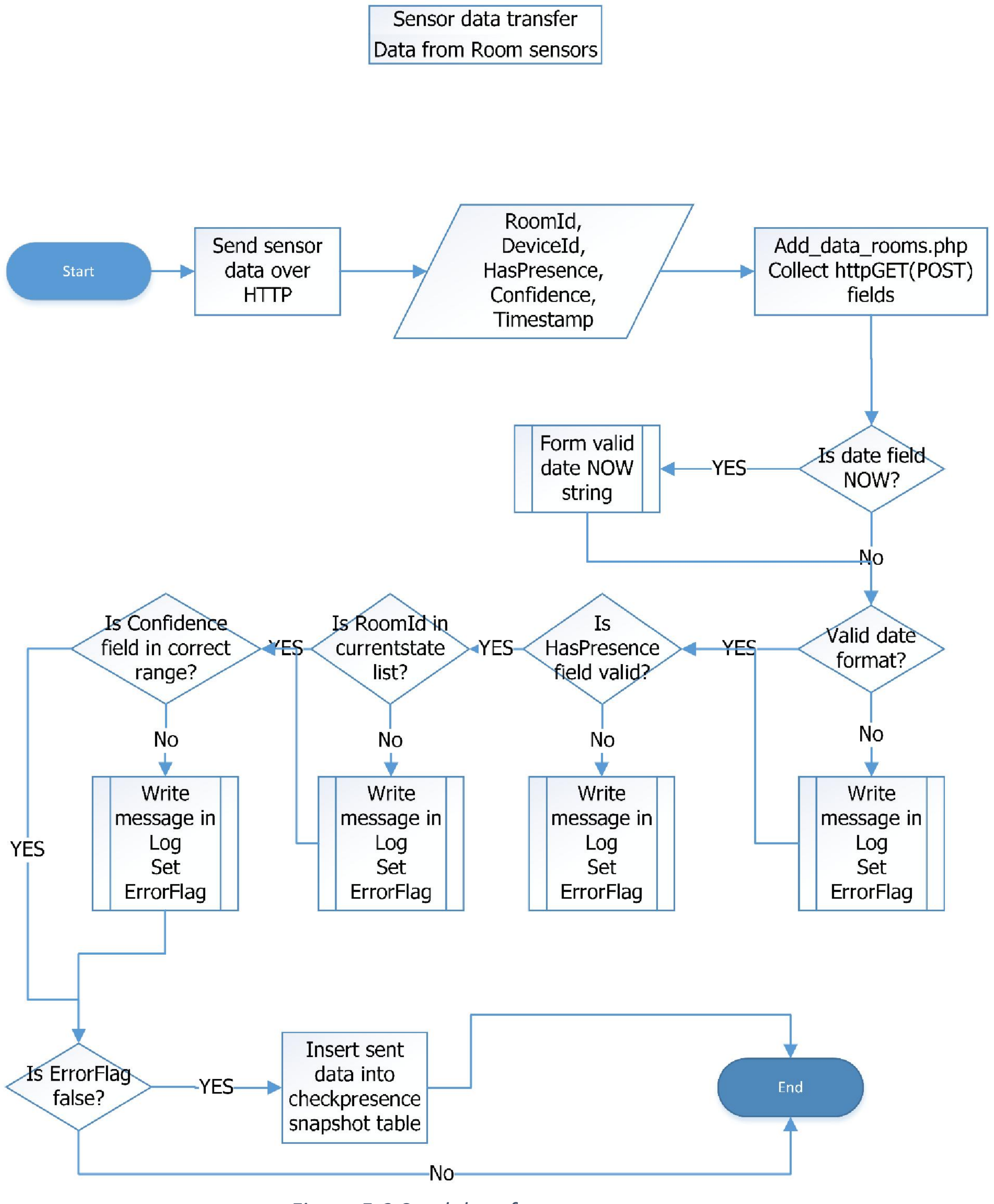
Sensor data transfer
Data from Room sensors
Start
Send sensor data over HTTP
RoomId, DeviceId, HasPresence, Confidence, Timestamp
Add_data_rooms.php Collect httpGET(POST) fields
Is date field NOW?
YES
Form valid date NOW string
No
Valid date format?
YES
No
Write message in Log Set ErrorFlag
Is HasPresence field valid?
YES
No
Write message in Log Set ErrorFlag
Is RoomId in currentstate list?
YES
No
Write message in Log Set ErrorFlag
Is Confidence field in correct range?
No
Write message in Log Set ErrorFlag
YES
Is ErrorFlag false?
YES
Insert sent data into checkpresence snapshot table
End
No

Figure 5-2 Send data from room sensors

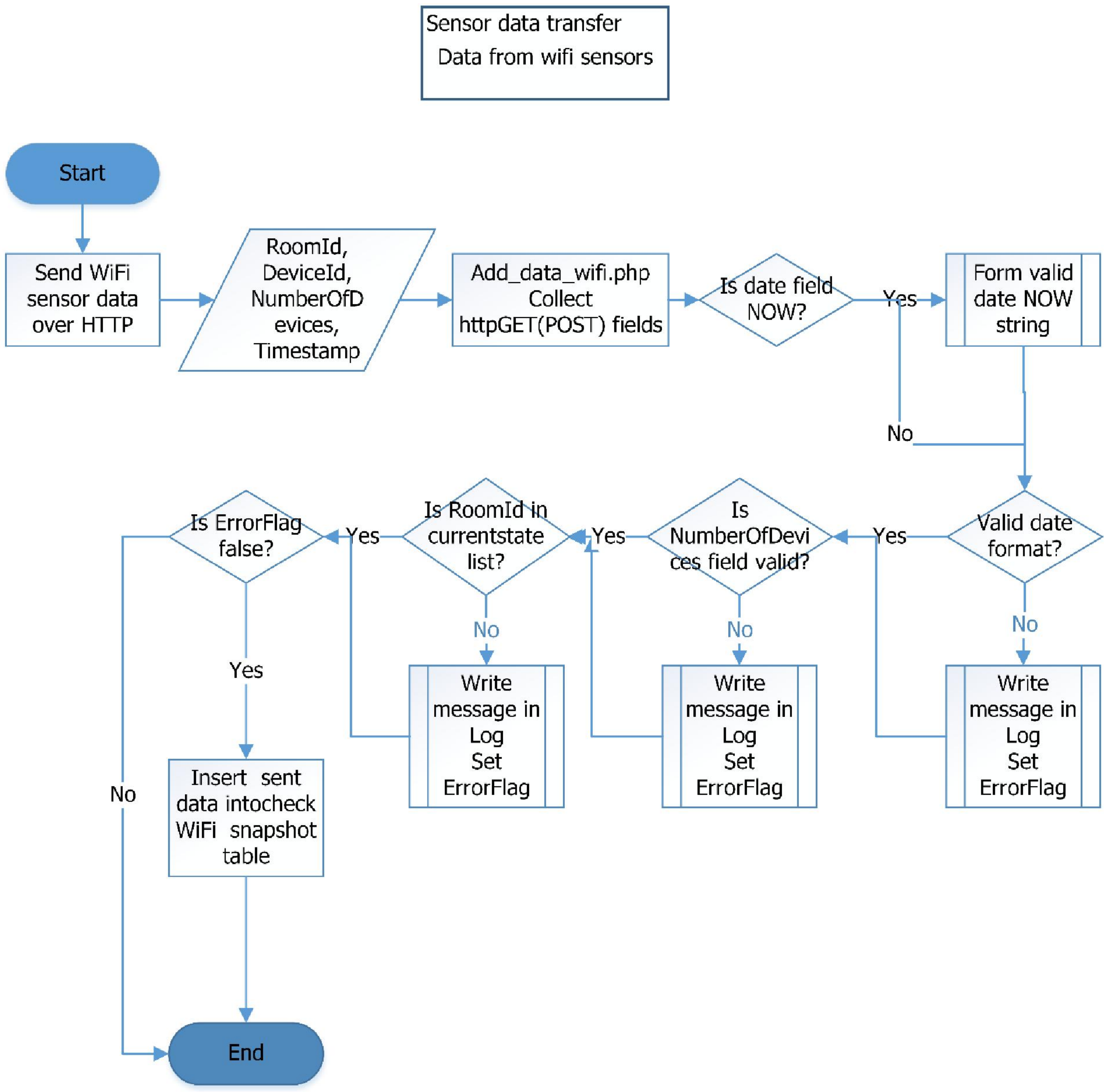
Sensor data transfer
Data from wifi sensors
Start
Send WiFi sensor data over HTTP
RoomId, DeviceId, NumberOfDevices, Timestamp
Add_data_wifi.php Collect httpGET(POST) fields
Is date field NOW?
Yes
Form valid date NOW string
No
Valid date format?
Yes
No
Write message in Log Set ErrorFlag
Is NumberOfDevices field valid?
Yes
No
Write message in Log Set ErrorFlag
Is RoomId in currentstate list?
Yes
No
Write message in Log Set ErrorFlag
Is ErrorFlag false?
Yes
Insert sent data intocheck WiFi snapshot table
No
End

Figure 5-3 Send data from Wi-Fi sensors

6. CLIENT SIDE FUNCTION FLOWS

The below diagram describes the normal flow of the QOE in client interaction part.

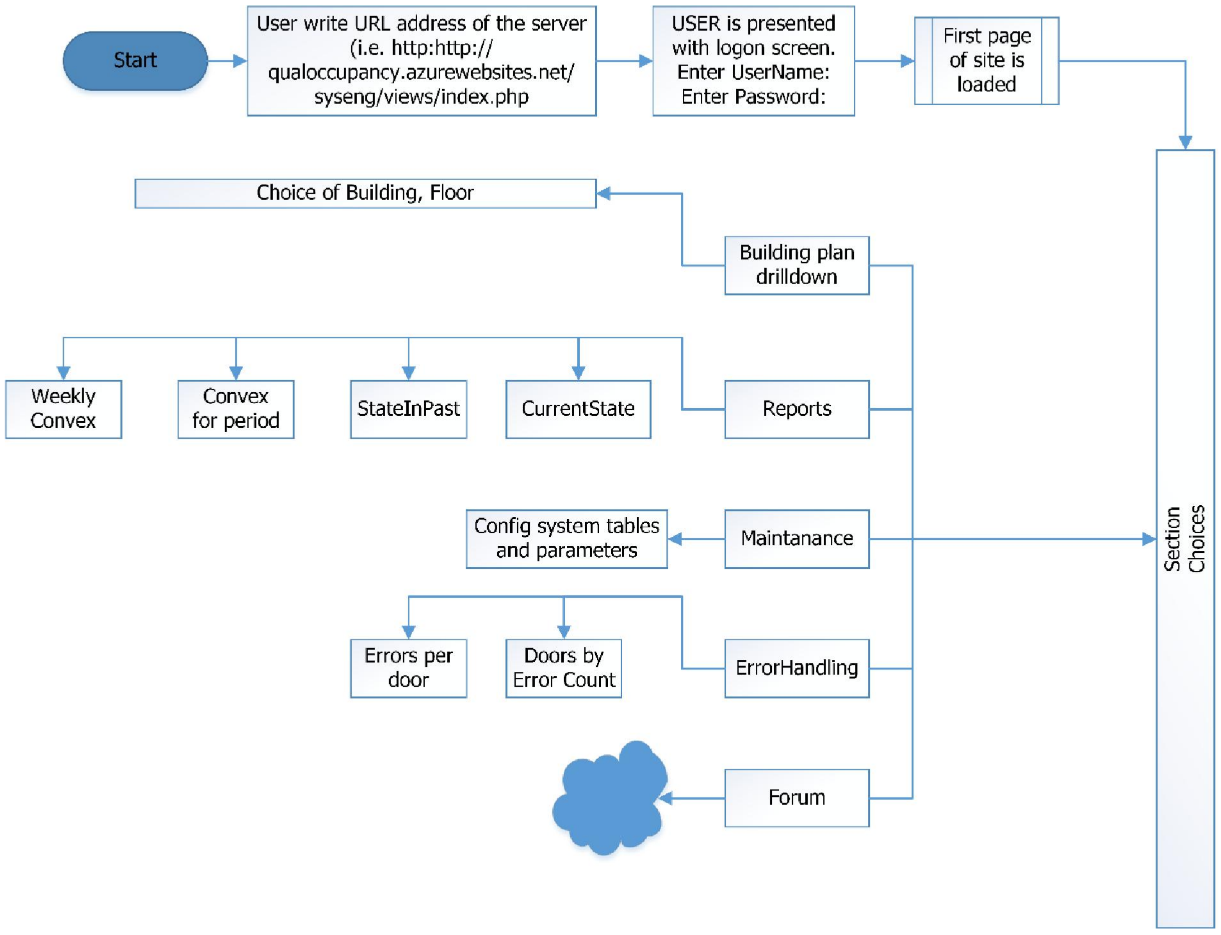
Start
User write URL address of the server (i.e. http:http:// qualoccupancy.azurewebsites.net/ syseng/views/index.php
USER is presented with logon screen. Enter UserName: Enter Password:
First page of site is loaded
Section Choices
Choice of Building, Floor
Building plan drilldown
Weekly Convex
Convex for period
StateInPast
CurrentState
Reports
Config system tables and parameters
Maintanance
Errors per door
Doors by Error Count
ErrorHandling
Forum

Figure 6-1 System use diagram

6.1. User Access Levels

After signing with username and password system keeps this session login info for all sections. No distinct user access levels are defined since under normal circumstances system is to be used by non-differentiated qualified users.

7. SYSTEM FUNCTIONALYTY IN DETAILS

7.1. System functionalities:

1. The system receives data sent from different sensors (using http protocol) that after format and logical validation is stored in the database.
   - from door sensors
   - from infrared room devices
   - from Wi-Fi devices
2. Accurately quantify the number of people in each room of a selected building, after each transition based on the information coming from different sensors.
3. Calculation of the confidence level of the number of people in every room, based on the confidence, received from the door sensors. When enough historical data is gathered the system could calculate the room confidence level from the collected historical data.
4. Error Handling Module - according to our client requirements, the system makes automatic correction of the errors.
   - The error checking and correction procedures are executed after every transmission of the data to ensure that we pick up any inconsistencies and errors on time. After checking the received data, the system makes automatically all necessary error corrections.
   - Another checking procedure is scheduled at night. On the base of information from door sensors and room devices. This procedure checks and compares occupancy in the rooms with the pre-defined set of rules. Using these rules, the system will be able to determine wrong activities in a room and correct them, tracing back to the entrance/exit of the building using breath first search algorithm. Later on, when there is enough data for an alyses, the system can create its own set of rules.
   - Checking procedure between the room sensor data and the door sensors data – making corrections where certain inconsistencies are found.
   - All automatic error corrections, made by the system, are stored in the correct error log table.

5. The room occupancy pattern – the occupancy convex procedure is used to find the pattern of the room occupancy - people presence in each room at certain times. **We** calculate the minimum, maximum and the average (mean) number of people in each room at a certain period of time (based on data for 10 weeks). Results are stored in a table and may be used for a smart error correction and reports base.
6. Responsive design User interface with:

- Building plan displays the real time data from the database – current number of people in every room.
- Reports for:
    - Current and past occupancy along with presence confidence level, with possibility for the user to select room, door, past period time interval.
    - Reports for occupancy convex (pattern)
        - Convex for a past period
        - Occupancy pattern for day of week. Graphical report allows drill down for periods.

7. Java Simulator for sending data from hardware sensors.
8. PHP module to create and load test samples of historical data.