AKN Public Services Project Software Design Document

Prepared By:

Ali Jaouhari

Kristina Wegener

Nadeen Alorainey

9/24/2014

Northern Kentucky University

Contents

1. INTRODUCTION

1.1 Purpose

This design document contains Iteration 2 and class diagrams of AKN Public Services System describing all of the classes and major interactions to be implemented in this iteration.

1.2 Scope

AKN Public Services System is a free web-based system developed by a group of NKU students to help individuals and organizations to look up and view open source data from the Open Source Cincy project.

1.3 Overview

This document will cover Iteration 2 for AKN Public Services System followed by the system domain model and system sequence diagrams .We also provide the data design ,human interface design and component design in the following parts.

2. ITERATION 2 PLAN

2.1 High Level Objectives:

+ Domain Model: Update the Domain Model to reflect any changes to the design

+ Vision: Update the Vision for any changes made during the iteration and refine the overall vision.

+ System Sequence Diagram: Update the System Sequence Diagram for all use cases for any changes made during the iteration.

+ Design Document: Update the Design Document to reflect all changes in the iteration.

2.2List of work items:

(Noted as issues in GitHub)

+ Domain Model: Kristi

+ System Sequence Diagram: Ali

+ Source Code: Kristi/Ali

+ Iteration Plan: Kristi

+ Vision: Kristi

+ Design Documentation: Kristi

+ Demonstration of Source Code: Kristi/Ali/Nadeen

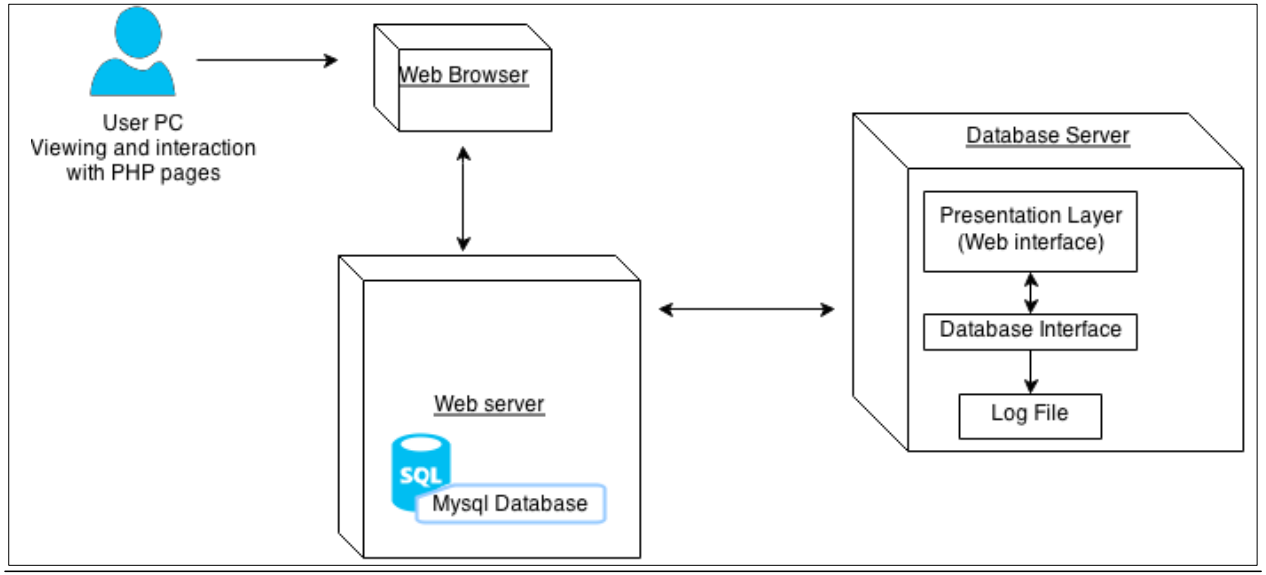
2.3Evaluation Criteria:

+ Testing source code

+ Demonstration of working application

3. SYSTEM ARCHITECTURE

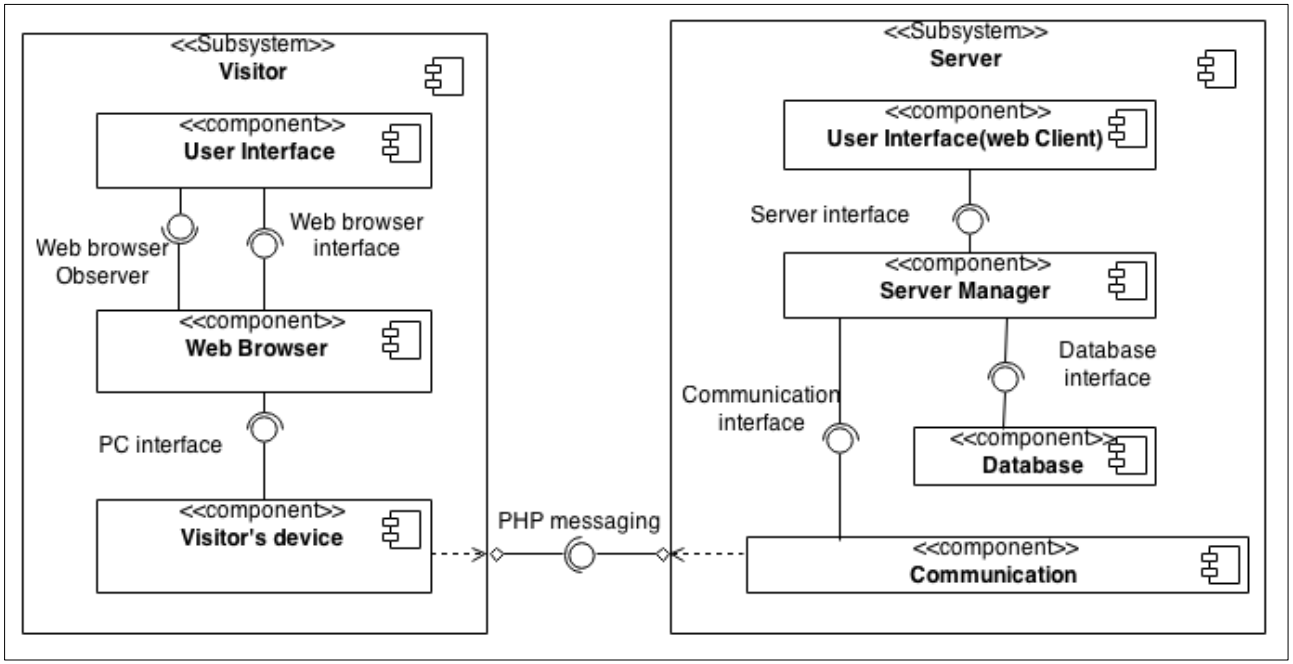
3.1 Architectural Design



The AKN Public Services System consists of one main system that runs on PHP. The system allows visitors to look up and access data stored on the server. The visitors will send requests for data to the server and receive requested data from the server and view information about the requested item.

The system will use PHP messaging to communicate. It will be capable of converting requests into PHP messages and parsing the PHP messages back into usable data.

3.2 Decomposition Description



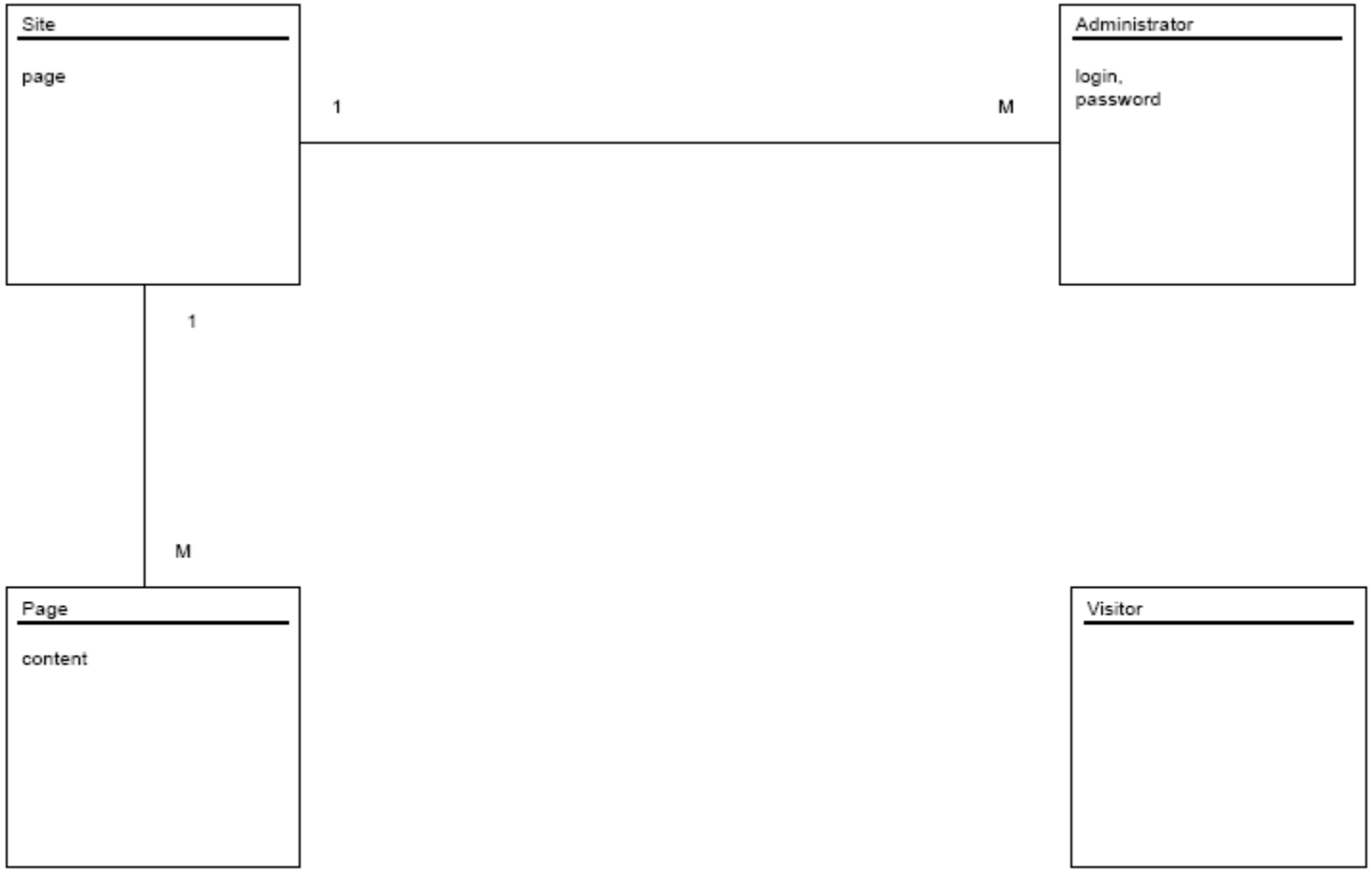
The visitor subsystem is divided up into a three layered architecture; it has a user interface, web browser, and visitor's device layer. Each layer has its own interface that other layers can use to interact with it. The user interface layer contains a web browser observer object and updates its data, using data from the observable application layer, via the observer pattern. The web browser layer handles threads, logs, and converts messages from the user interface layer into PHP messages and sends them to the device layer. The device layer handles the interactions with the hardware. Each layer is also a component that can be individually updated or replaced as long as the interface remains the same.

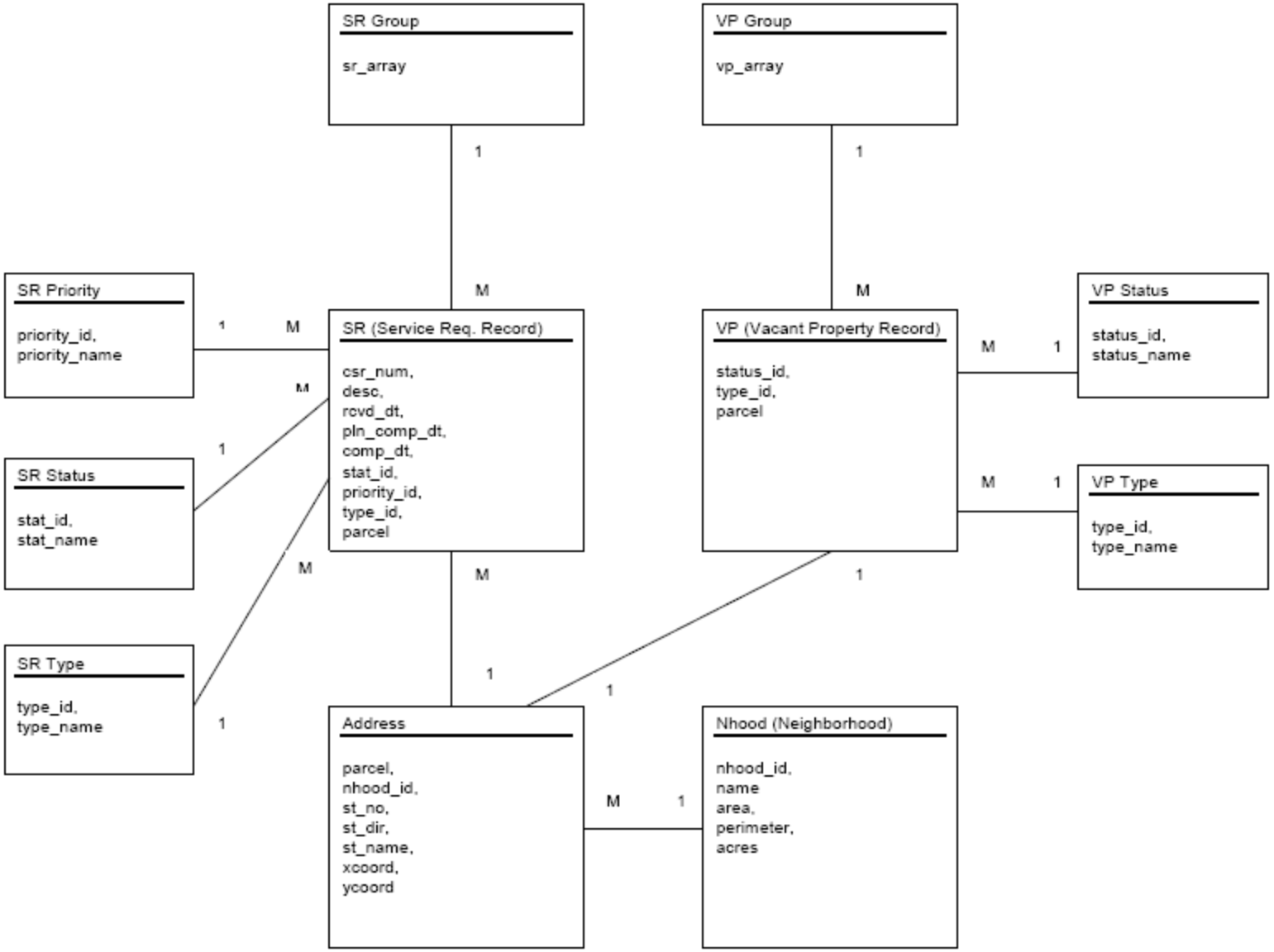
The server subsystem contains a web client component, server manager component, database component, and communication component. The web client component is a simple desktop application that connects to the server and follows the same rules for communication as the visitor system; it sends and receives PHP messages. The server manager component handles the threads, message parsing, and database queries. The database component stores all of the data for the system. The communication component handles the ports and the sending and receiving of messages.

3.3Design Class Diagram

4. DATA DESIGN

4.1 Domain Model




SR Group
sr_array
VP Group
vp_array
SR Priority
priority_id,
priority_name
SR (Service Req. Record)
csr_num,
desc,
rcvd_dt,
pln_comp_dt,
comp_dt,
stat_id,
priority_id,
type_id,
parcel
VP (Vacant Property Record)
status_id,
type_id,
parcel
VP Status
status_id,
status_name
SR Status
stat_id,
stat_name
VP Type
type_id,
type_name
SR Type
type_id,
type_name
Address
parcel,
nhood_id,
st_no,
st_dir,
st_name,
xcoord,
ycoord
Nhood (Neighborhood)
nhood_id,
name
area,
perimeter,
acres
1
M

4.2 Data Dictionary

The information domain of our system consists of the different types of information about Service Request, and Vacant Property records. The data stored in our central database is queried by the server which receives messages from the web-app. The data is organized by tables in a relational database.

- SR (Service Request Record):
  - csr_num: the number of the Service Request.
  - desc: the Service Request's written description.
  - rcvd_dt: the date the Service Request was placed.
  - pln_comp_dt: the date the Service Request should be completed.
  - comp_dt: the date the Service Request was completed.
  - stat_id: an id number for the status of the Service Request.
  - priority_id: an id number for the priority of the Service Request.
  - type_id: an id number for the type of Service Request.
  - parcel: the parcel number for the location of the Service Request.
- SR Type (Service Request type):
  - type_id: an id number for the type of the Service Request record.
  - type_name: The type of the Service Requested.
- SR Priority (Service Request priority):
  - priority_id: id number for the priority of the requested service.
  - priority _name: The priority of the Service Requested.
- SR Status (Service Request Status)
  - stat_id: an id number for the status of the Service Request record.
  - stat _name: The status of the Service Request record.
- SR Group:
  - sr_array: array of Service Request records.
- VP (Vacant Property Record):
  - stat_id: an id number for the Vacant Property record
  - type_id: an id number for the Vacant Property type.
  - parcel: the parcel number for the location of the Vacant Property.

- VP status (Vacant Property Status):
  - status_id: an id number for status of the Vacant Property.
  - status_name: status of the Vacant Property.
- VP Type (Vacant Property Status):
  - type _id: an id number for the Vacant Property type.
  - type_name: The type of Vacant Property.
- Address:
  - parcel: the parcel number of the property.
  - nhood_id: an id number for the neighborhood .
  - st_no: the number of the street.
  - st_name: the name of the street.
  - st_dir: the street direction (ex: North, South, East West. Optional)
- Nhood (Neighborhood)
  - name: The name of the neighborhood
  - nhood_id: an id number for the neighborhood.
  - area: the area for the neighborhood
  - perimiter: the perimeter of the neighborhood in miles.
  - acres: amount of acreage in a neighborhood.
- Administrator
  - login: The name of the administrator.
  - password: The password for the administrator login.

5. HUMAN INTERFACE DESIGN

5.1 Overview of User Interface

The flow of the interface has been designed around the idea of searching for records, and displaying the returned. After displaying the information returned an option will be available at the bottom of the list to display information about that record on a map. After selecting display on map, the system will load another page with map view and the information displayed on it.

5.2 Screen Images