**Software Design Document**

**and Programmers Guide**

**CS Virtual Tour**

**06/03/2014**

**Revision History**

|  |  |  |
| --- | --- | --- |
| **Revision** | **Description** | **Date** |
| Original | Original Release | 06/03/2014 |
| 1.01 | Revised Architecture Diagram | 06/11/2014 |
| 1.1 | Updated changes that occurred throughout the development process | 12/05/2014 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Table of Contents**

[1. Introduction](#h.30j0zll)

[2. Overview of the System](#h.1fob9te)

3. Architecture

[4. Interface Design](#h.2et92p0)

[5. Component Design](#h.tyjcwt)

[6. Appendices](#h.3dy6vkm)

# Introduction

The purpose of this document is to describe the technical design of the WWU Computer Science Dept Virtual Tour suite.

# Overview of the System

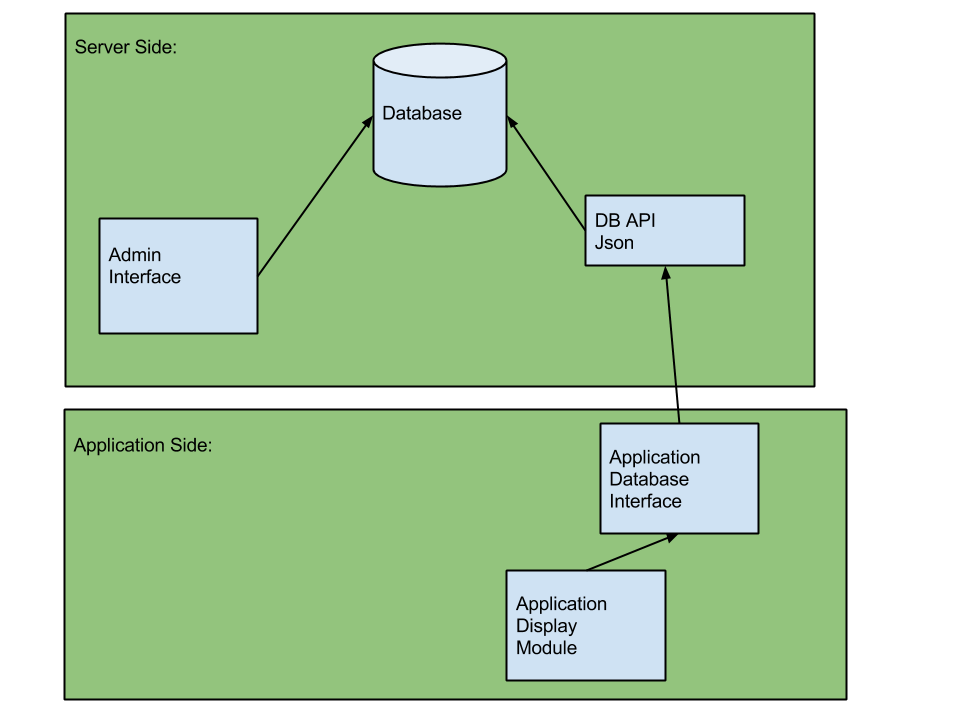
The system is mobile application that connects to a database via a web API to fetch and display content concerning the WWU Computer Science deptartment. Tour stop content is stored as different components for text, image, and video. The components are passed to the application as JSON objects, and the application uses these to construct a linear layout.

The application is accompanied by a web based administrative interface. This interface allows the tour administrators to add, remove, and edit tour stop content by manipulating individual components. The interface also allows to assign stops to a corresponding location on the map, and associate QR identifiers with each stop.

# Architecture

The system is comprised of four main parts. The Admin Interface, the Database, the Database API, and the Application. The Application is further subdivided into database interface and display modules. The Admin Interface, the Database, and the Database API shall all reside on the same server. The primary purpose of the Database API is to limit access to the database so that a compromised client cannot execute malicious or ill formed queries on the database. It also serves to construct queries and convert the results into generic JSON responses so the application is not required to know the details of the database implementation.

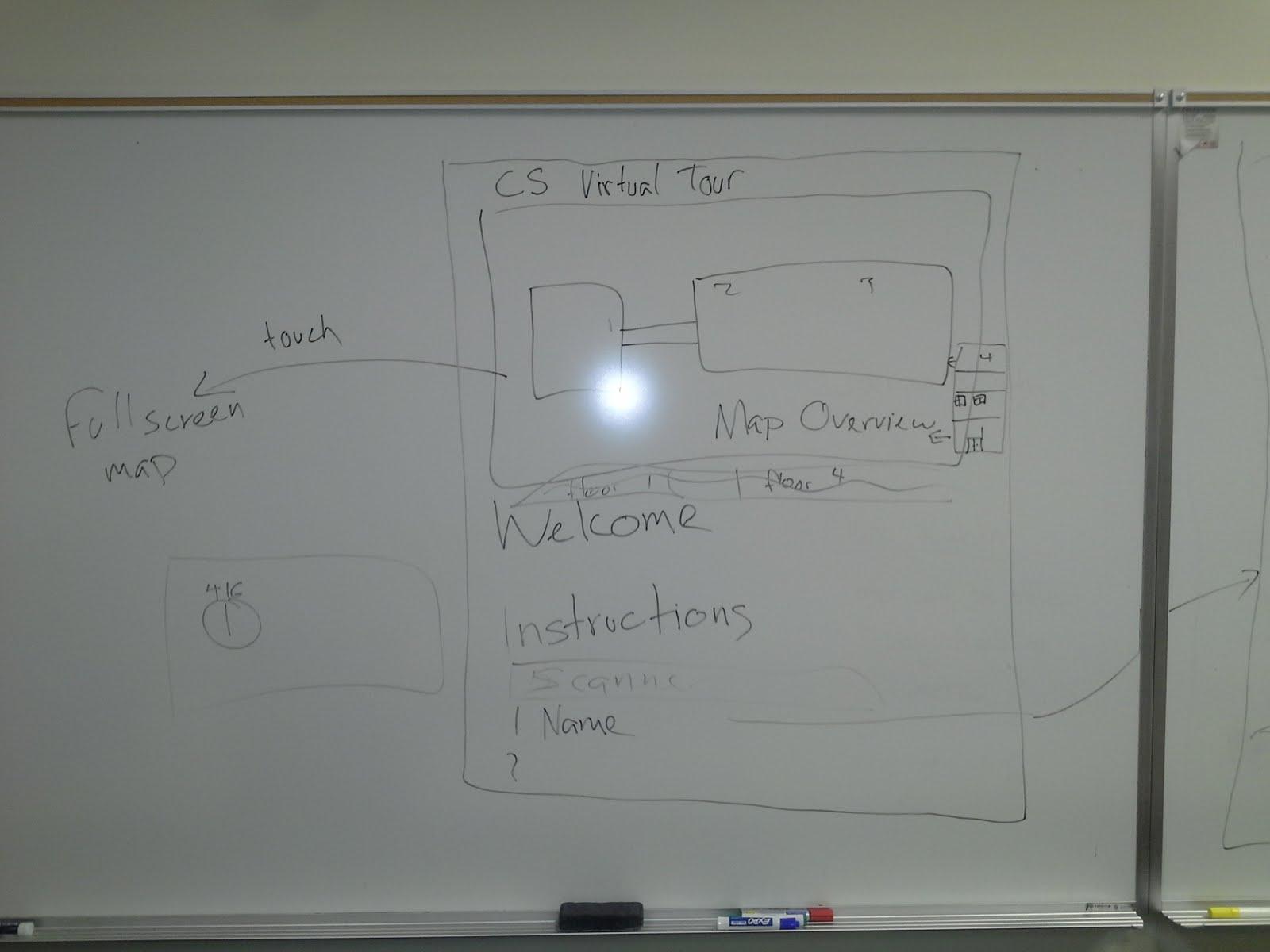
**System Architecture Diagram:**

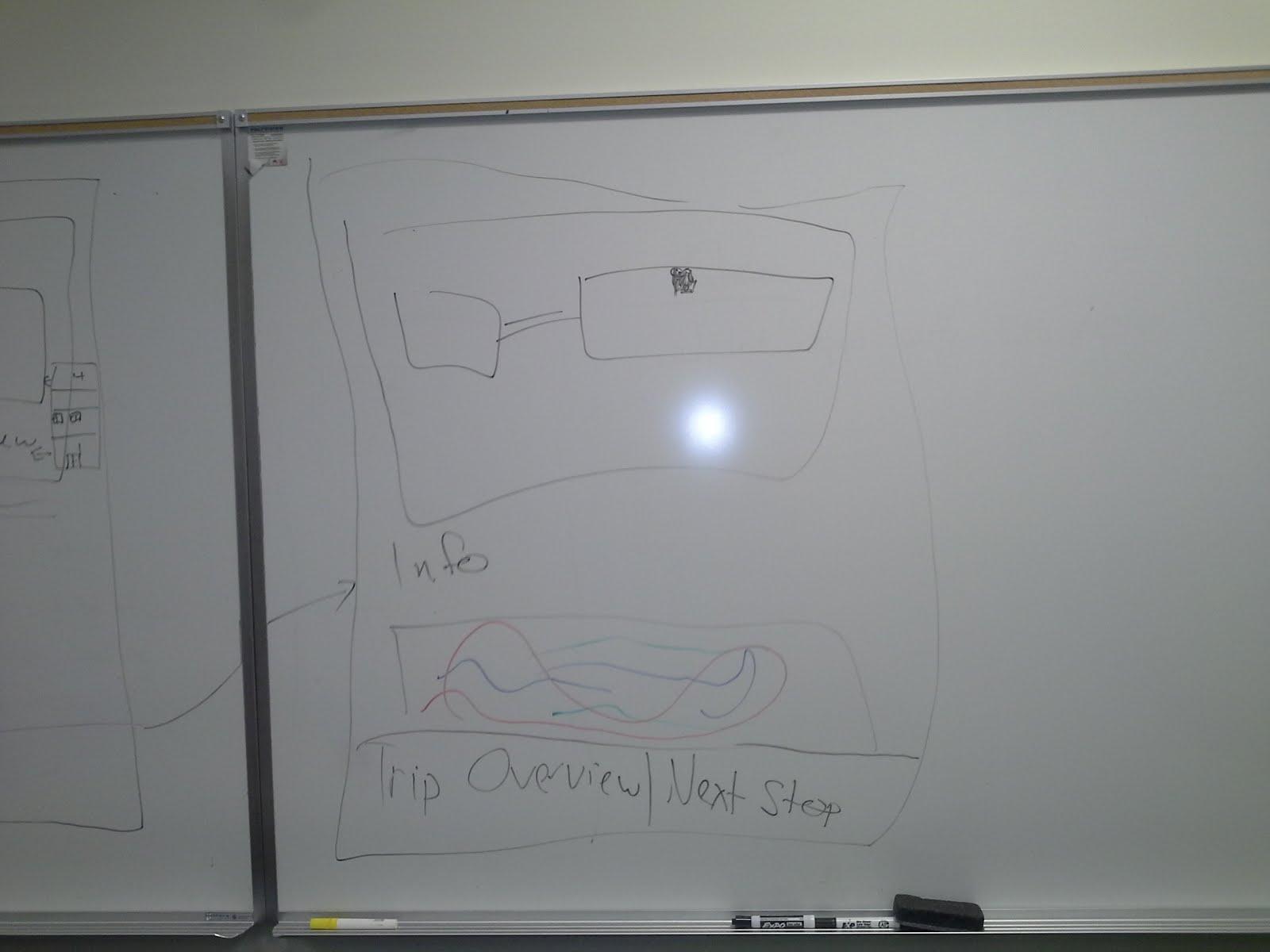


# Interface Design

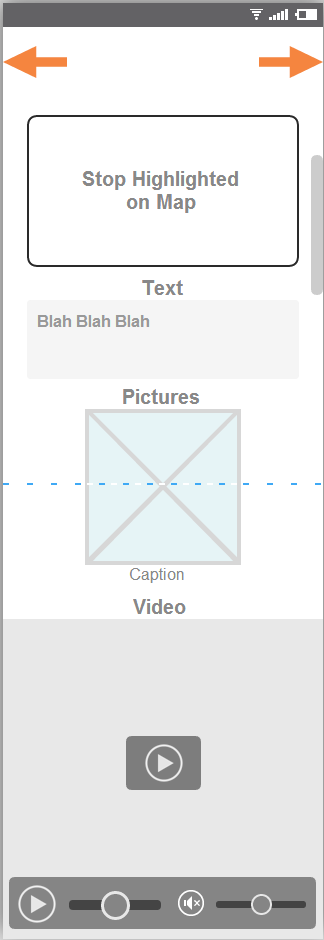
The interface is designed to be dynamically generated for each tour stop. This allows the tour content to be changed with minimal or no changes to the application interface.

**Interface mockups from customer meeting:**

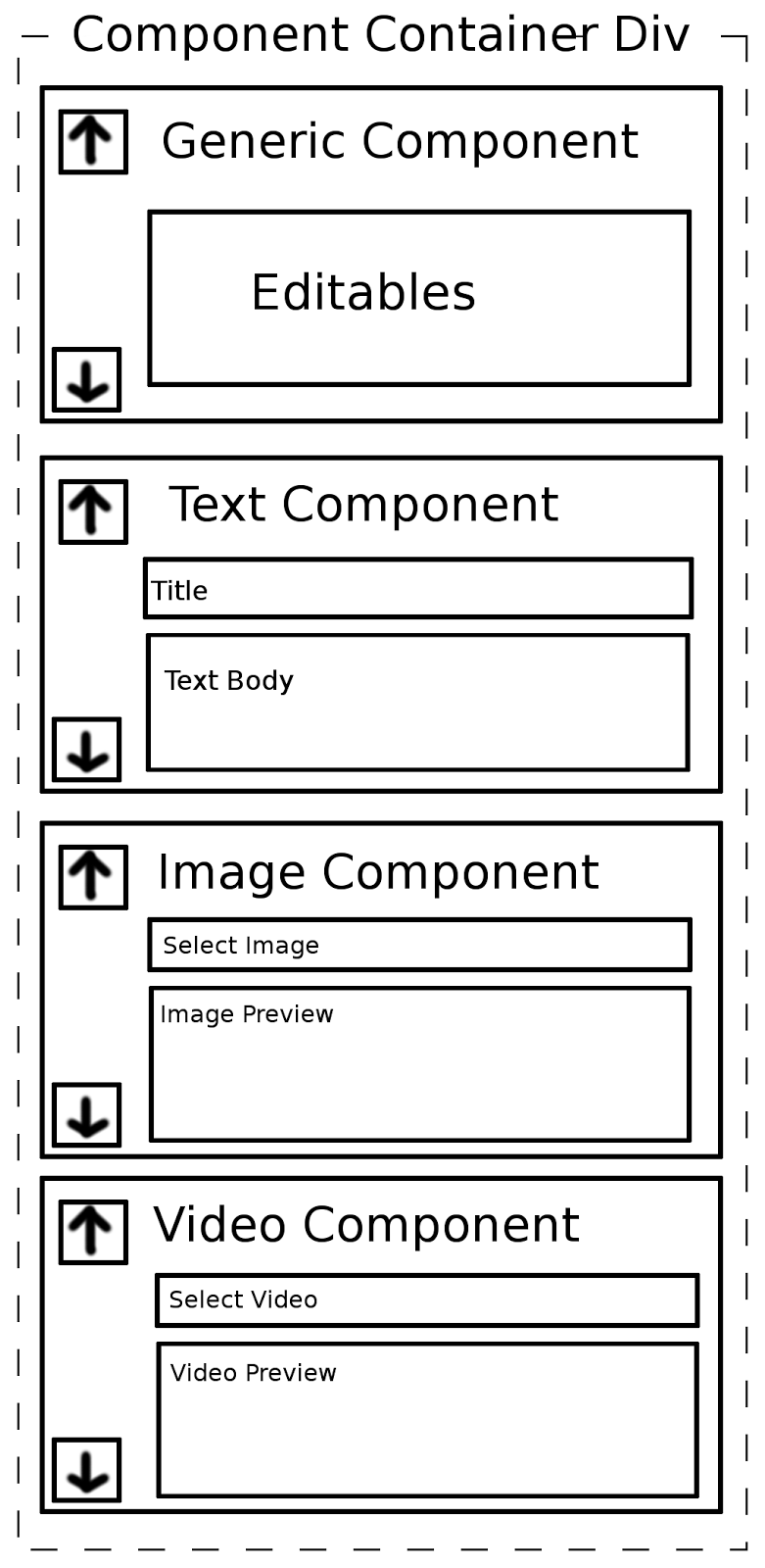




**Application Wireframe mock ups**



**Priorities 32, 34, 42, 46 - Admin interface component framework wireframes:**



# Component Design

### Server side database api:

**Sequence Diagram:**

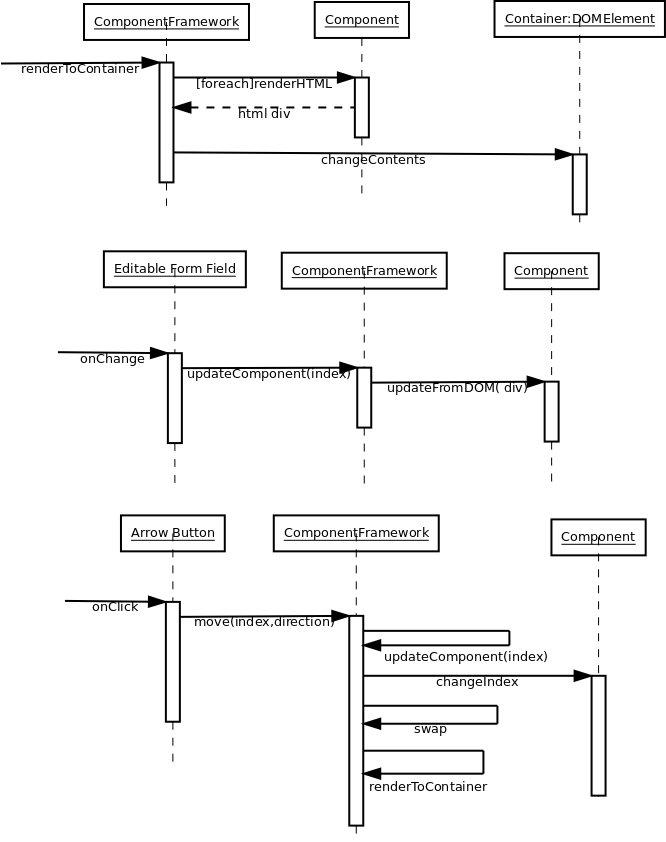
### DataAPI-ServerSide.png

### 

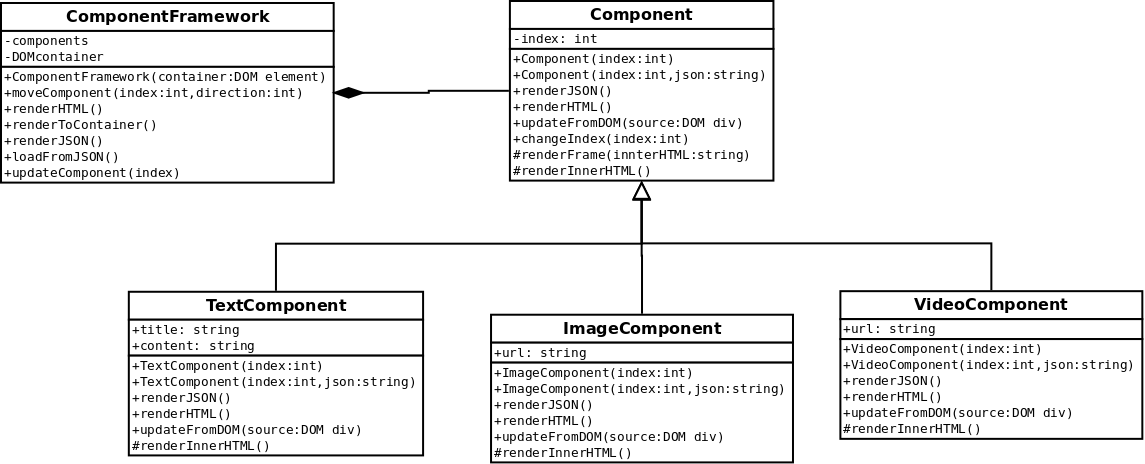
### 

### Admin Interface Component Editing Framwork - Priorities 32, 34, 42,46:

**Sequence Diagram:**



**Class Diagram:**



# Appendices

**Database Schema:**

SoftwareSoftware

**Database API Message Format:**

The database API return all results as JSON objects conforming to the following specification:

|  |  |
| --- | --- |
| **Field** | **Purpose** |
| success | true/false, indicates whether the request was successful |
| result | A JSON object containing the contents of the entry in the Stops table corresponding to the requested stop id |
| error | A string describing any errors that occurred (if any) |
| stoplist | A JSON array containing partial records for all entries in the Stops table, passed only if the requested stop id was 0 |

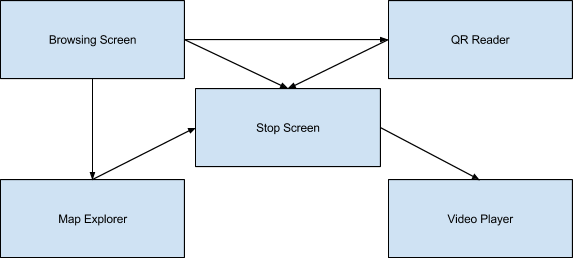
**Tour Stop Content Markup Format:**

Tour stop content is made of a linear arrangement of three different types of components: text, image, and video. The components are stored as an array of JSON components in the database. The components have the following properties:

|  |  |  |
| --- | --- | --- |
| **Component Type** | **Field** | **Description** |
| Text | Title | Descriptive title of the text section |
| Text | Content | Contents of the text section |
| Image | Title | Title caption for the image |
| Image | URL | Absolute url of the image file |
| Video | Title | Title caption for the video |
| Video | URL | Absolute url of the video file |

Application Programmers Guide

# Flow



# Structures

1. **Stop.java**
   1. Class containing all the data fields of a Stop. Also implements Comparable, Stops are compared based on their stopOrder.
2. **Map.java**
   1. Class containing all the data fields of a Map. Also implements Comparable, Maps are compared based on their mapOrder,

# Augmented Views

1. **MapImageView.java**
   1. An ImageView that displays a map with stops associated with that map marked upon it. Can also have a selected stop that will be drawn a different color from the rest.
2. **MapTouchImageView.java**
   1. A zoomable and scrollable Imageview with stops associated with a map marked upon it. Stops are labeled by their stopRoomNumber.

# Activities

1. **MainActivity.java**
   1. The home activity of the application. Implements functionality for navigation to various other parts of the application.
2. **StopActivity.java**
   1. Activity that displays information for a specific stop. Implements functionally for navigation between stops.
3. **ImageViewActivity.java**
   1. Activity that displays a scroll/zoom-able image.
4. **VideoPlayerActivity.java**
   1. Activity that streams a video file with media controls.
5. **QRReaderActivity.java**
   1. Activity that uses the camera is scan QR Codes, which then navigates to the specified stop.

# Helper Classes

1. **CameraPreview.java**
   1. Generates a preview of the camera.
2. **ImageProcessor.java**
   1. Resizes large images before they are used to improve performance.
3. **ImageRetrievalTask.java**
   1. Retrieves images from the web.
4. **MapRetrievalTask.java**
   1. Retrieves the map list from the server specified at MAP\_URL.
5. **MapSpinnerAdapter.java**
   1. Basic spinner adapter for selecting maps on the main screen.
6. **StopRetrievalTask.java**
   1. Retrieves a list of stops or an individual stop from the server specified at STOP\_URL.
7. **ThumbnailRetrievalTask.java**
   1. Retrieves thumbnail previews for videos located at remote locations.

# Interfaces

1. **OnContentLoaded.java**
   1. Specifies when a piece of asynchronous content has been loaded.
2. **OnTaskCompleted.java**
   1. Specifies when Stop or Map data has been retrieved from the server.

# External Libraries

1. **zbar** 
   1. QR reader library used in our application.
2. **FFMPEGMediaMetaDataRetriever**
   1. Library for retrieving remote media meta data such as video thumbnails.
3. **TouchImageView**
   1. Implementation of an ImageView that is capable of scrolling and pinch to zoom.