|  |
| --- |
|  |
| Capstone Project Document |

**First Aid**

Report #2 – Architecture Design

|  |  |  |
| --- | --- | --- |
| **First Aid VN** | | |
| **Group Members** | Đàm Huy Hùng | SE02585 |
| Nguyễn Duy Anh | 01525 |
| Mai Trung Kiên | SE02994 |
| Nguyễn Tiến Dũng | SE03272 |
| Nguyễn Phúc An | SE03118 |
| **Supervisor** | Mr. Nguyễn Văn Sang | |
| **Project code** | FAVN | |

**- Hanoi, 09/2016 -**

# SIGNATURE PAGE

AUTHOR: Mai Trung Kiên 28/09/2016

Technical Leader

REVIEWERS: Đàm Huy Hùng 30/09/2016

Project Manager

APPROVAL: Nguyễn Văn Sang --/--/--16

Supervisor

Record of change

\*A - Added M - Modified D – Deleted

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Effective Date | Changed Item | A,M,D | Change Description | Reason for Change | Rev. Number |
| 28/Sep/2016 | Create Architecture Design | A | First version | Create Architecture Design | 1.0 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

TABLE CONTENTS

[SIGNATURE PAGE 0](#_Toc469215921)

[1 PROJECT OVERVIEW 4](#_Toc469215922)

[1.1 Purpose 4](#_Toc469215923)

[1.2 Scope 4](#_Toc469215924)

[1.3 Definitions, Acronyms, Abbreviations 4](#_Toc469215925)

[1.4 References 4](#_Toc469215926)

[1.5 Overview 4](#_Toc469215927)

[2 CHOICE OF ARCHITECTURE DESIGN 5](#_Toc469215928)

[2.1 MVC MODEL 5](#_Toc469215929)

[2.1.1 MVC Model Overview 5](#_Toc469215930)

[2.1.2 Advantages and disadvantages of MVC Model 6](#_Toc469215931)

[2.1.3 The reasons for choosing MVC Model 6](#_Toc469215932)

[2.2 Android 7](#_Toc469215933)

[2.2.1 Android Overview 7](#_Toc469215934)

[2.2.2 Advantages and disadvantages of Android platform 7](#_Toc469215935)

[2.2.3 The reasons for choosing Android platform 7](#_Toc469215936)

[2.3 Laravel 8](#_Toc469215937)

[2.3.1 Laravel Overview 8](#_Toc469215938)

[2.3.2 Advantages and disadvantages of Laravel 9](#_Toc469215939)

[2.3.3 The reason for choosing Laravel 9](#_Toc469215940)

[3 ARCHITECTURAL REPRESENTATION 10](#_Toc469215941)

[3.1 First Aid Application 10](#_Toc469215942)

[3.2 Ambulance Application 10](#_Toc469215943)

[3.3 Dispatcher Application 11](#_Toc469215944)

[3.4 Admin/Expert Application 12](#_Toc469215945)

[4 ARCHITECTURAL GOALS AND CONSTRAINTS 12](#_Toc469215946)

[5 Use-case View 13](#_Toc469215947)

[6 Logical View 13](#_Toc469215948)

[6.1 Overview 13](#_Toc469215949)

[6.2 Architecturally Significant Design Packages 14](#_Toc469215950)

[6.2.1 First Aid Application 14](#_Toc469215951)

[6.2.2 Ambulance Application 15](#_Toc469215952)

[6.2.3 Dispatcher Application 16](#_Toc469215953)

[6.2.4 Admin/Expert Application 18](#_Toc469215954)

[7 Process view 20](#_Toc469215955)

[7.1 Android app module process view 20](#_Toc469215956)

[7.2 Web module process view 21](#_Toc469215957)

[8 Deployment View 21](#_Toc469215958)

[8.1 Android module deployment view 21](#_Toc469215959)

[8.2 Web module deployment view 22](#_Toc469215960)

[9 Quality 22](#_Toc469215961)

# PROJECT OVERVIEW

## Purpose

This document describes system overview of FAVN (first aid and emergency assistance services) system. It uses view of architectures to describe the system. It was designed as a basis for choosing the appropriate architecture for building this systems.

## Scope

The scope of this document is to depict the architecture of the FAVN system created by FAVN capstone project team.

## Definitions, Acronyms, Abbreviations

|  |  |  |
| --- | --- | --- |
| **Acronym** | **Definition** | **Note** |
| FAVN | First Aid Vietnam |  |
| DB | Database |  |
| MVC | Model view control |  |
| IDE | Integrated development environment |  |
| Q&A | Question and answer |  |
| GUI | Graphic user interface |  |
| REST | Representational state transfer |  |
| API | Application Programming Interface |  |

Table ‑: Definitions and Acronyms.

## References

* FAVN\_ Software Requirements Specification\_v1.0\_EN.docx
* FAVN\_Data Design\_v1.0\_EN.docx
* <https://www.tutorialspoint.com/>
* https://en.wikipedia.org
* Software Architecture Design Illuminated Book

## Overview

The Software Architecture Document contains the following subsections:

* **Section 1**: Provide an overview of entire Software Architecture Document.
* **Section 2**: Choice of Architecture Design
* **Section 3**: Architectural Representation
* **Section 4**: Architectural Goals and Constraints
* **Section 5**: Use-Case view
* **Section 6**: Logical View
* **Section 7**: Process View
* **Section 8**: Deployment view
* **Section 9**: Quality

# CHOICE OF ARCHITECTURE DESIGN

## MVC MODEL

The purpose of FAVN is developing a system of first aid supporting and other emergency features. The system of FAVN is structured based on MVC combined with layered architecture.

### MVC Model Overview

The **model-view-controller or MVC** is software architecture commonly used for creating web or software applications. In other words, it's a structure for applications to follow in order to ensure efficiency and consistency.

The Model-View-Controller (MVC) architectural pattern separates an application into three main components: the model, the view, and the controller. The model consists of application data, business rules, logic and functions. A view can be any output representation of data. The controller mediates input, converting it to commands for the model or view.

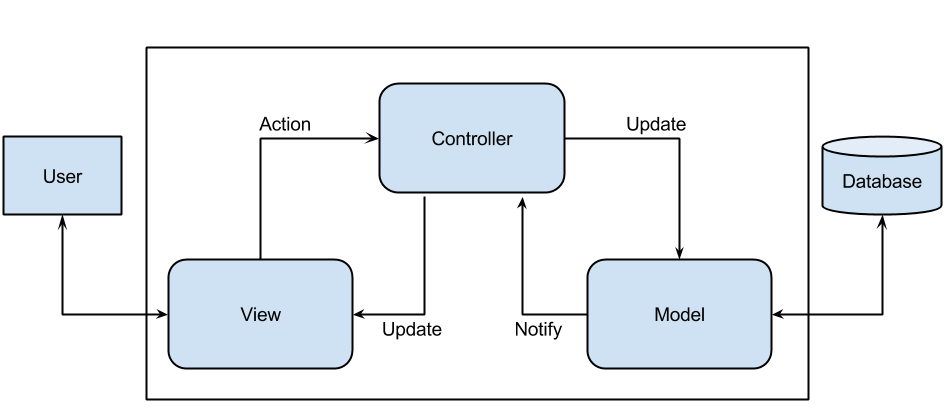


Figure 2-1: MVC Model.

More detail about each component is divided by the MVC pattern:

* **Model:** Model represents an object carrying data. It can also have logic to update controller if its data changes.
* **View:** View represents the visualization of the data that model contains.
* **Controller:** Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

### Advantages and disadvantages of MVC Model

* Advantages:
* **Faster development process:** MVC supports rapid and parallel development.
* **Ability to provide multiple views:** In the MVC Model, you can create multiple views for a model.
* **Support for asynchronous technique:** MVC also supports asynchronous technique, which helps developers to develop an application that loads very fast.
* **Modification does not affect the entire model:** Modification does not affect the entire model because model part does not depend on the views part.
* **MVC model returns the data without formatting:** MVC pattern returns data without applying any formatting so the same components can be used and called for use with any interface.
* Disadvantages:
* **Time consuming in small project development process:** for small projects that apply MVC model caused cumbersome.
* **Increased complexity.**
* **Not suitable for agent-oriented applications:** such as interactive mobile and robotics applications.
* **Data model change expensive:** multiple pairs of controllers and views based on the same data model.
* **Knowledge on multiple technologies is required.**

### The reasons for choosing MVC Model

* MVC makes parts of system can be developed individually and simultaneously to reduce developing time.
* It is good support for application built by project team has many developers and designers but still managed application features.
* Better support for test-driven development.
* Tools is useful and documentary source is large makes MVC is easy to develop.
* FAVN system is not complete system, now. We built the system that towards extensibility and maintainability in the future.

## Android

### Android Overview

Android is an open source and Linux-based Operating System for mobile devices such as smartphones and tablet computers. Android was developed by the Open Handset Alliance, led by Google, and other companies.

Android offers a unified approach to application development for mobile devices which means developers need only develop for Android, and their applications should be able to run on different devices powered by Android.

### Advantages and disadvantages of Android platform

* Advantages
* Open source.
* Larger developer and community reach.
* Increased marketing.
* Inter app integration.
* Reduced cost of development.
* Higher success ratio.
* Rich development environment.
* Disadvantages
* Multitude of devices to take care of, and not just different screen sizes, but different resolutions too. This makes the app design, as well as UI development much harder.
* Quality of apps on the Play Store is much worse than those on the App Store. This is primarily because of Apple's stringent review policies. They do not allow low quality apps to be published.

### The reasons for choosing Android platform

* Most people uses android handheld devices especially android phone.
* Android is a powerful operating system and supports great features.
* There are many plugins, resources and documents which support creating an android application.

## Laravel

### Laravel Overview

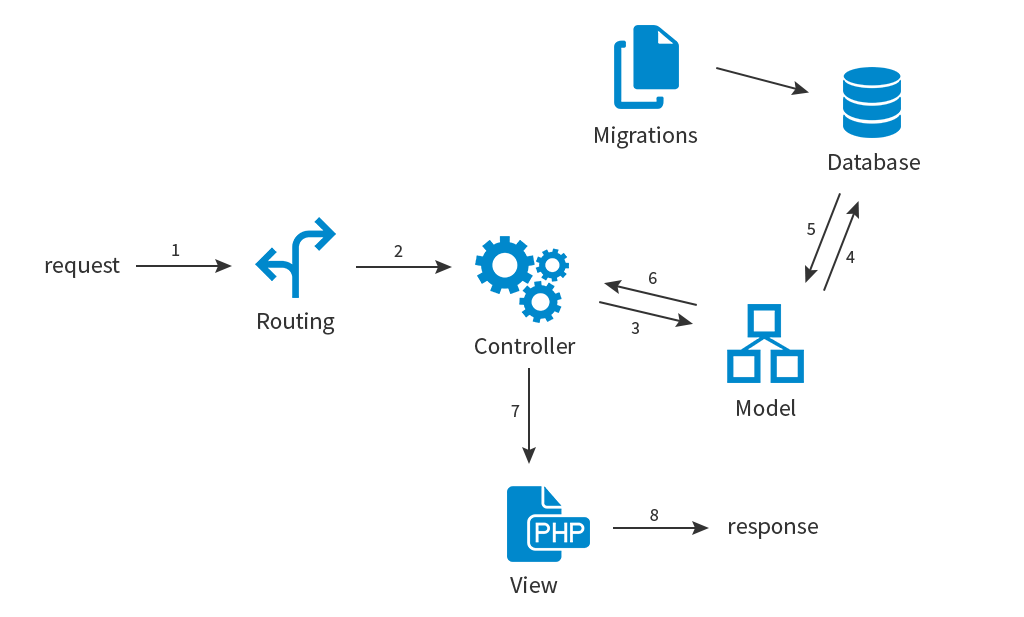
Laravel is referred to as a “full stack” framework because it handles everything from web serving to database management right down to HTML generation. A vertically integrated web development environment can provide a better experience for the developer.

Figure -2: Architecture of Laravel.

* **Features:**
* Bundles provide a modular packaging system, with bundled features already available for easy addition to applications. Furthermore, Laravel uses Composer as a dependency manager to add framework-agnostic and Laravel-specific PHP packages available from the Packagist repository.
* Eloquent ORM (object-relational mapping) is an advanced PHP implementation of the active record pattern, providing at the same time internal methods for enforcing constraints on the relationships between database objects. Following the active record pattern, Eloquent ORM presents database tables as classes, with their object instances tied to single table rows.
* Query builder, provides a more direct database access alternative to the Eloquent ORM. Instead of requiring SQL queries to be written directly, Laravel's query builder provides a set of classes and methods capable of building queries programmatically. It also allows selectable caching of the results of executed queries.
* Application logic is an integral part of developed applications, implemented either by using controllers or as part of the route declarations.
* Reverse routing defines a relationship between the links and routes, making it possible for later changes to routes to be automatically propagated into relevant links.
* Restful controllers provide an optional way for separating the logic behind serving HTTP GET and POST requests.
* Class auto loading provides automated loading of PHP classes without the need for manual maintenance of inclusion paths.
* View composers serve as customizable logical code units that can be executed when a view is loaded.
* Blade templating engine combines one or more templates with a data model to produce resulting views.
* Migrations provide a version control system for database schemas, making it possible to associate changes in the application's codebase and required changes in the database layout.
* Unit testing is provided as an integral part of Laravel.
* Automatic pagination simplifies the task of implementing pagination, replacing the usual manual implementation approaches with automated methods integrated into Laravel.
* Form request is a feature of Laravel 5 that serves as the base for form input validation by internally binding event listeners, resulting in automated invoking of the form validation methods and generation of the actual form.

### Advantages and disadvantages of Laravel

* Advantages:
* It uses a blade template engine to speed up compiling tasks, and users can include latest features so easily.
* “Bundled modularity” enables code reusing without much hassle.
* An outstanding Artisan CLI comprising advanced tools to do tasks, and migrations.
* Splendid documentation, and an added feature of reverse routing.
* Disadvantages:
* Amateur developers face problems while extending codes and classes.
* Many methods included in the reverse routing are complex.

### The reason for choosing Laravel

* **Open Source:** It is a free open-source PHP web application framework which has easy process of building complex and large web applications with code maintainability. Provide a more fluid user experience akin to a desktop application.
* **MVC Architecture Support:** It follows the MVC pattern, ensuring the transparency between logic and presentation.
* **Libraries and configuration:** It supports many development environments and adjusts itself based on the appropriate platform where app is running.

# ARCHITECTURAL REPRESENTATIOn

We follow MVC architecture to implement the FAVN system.

## First Aid Application

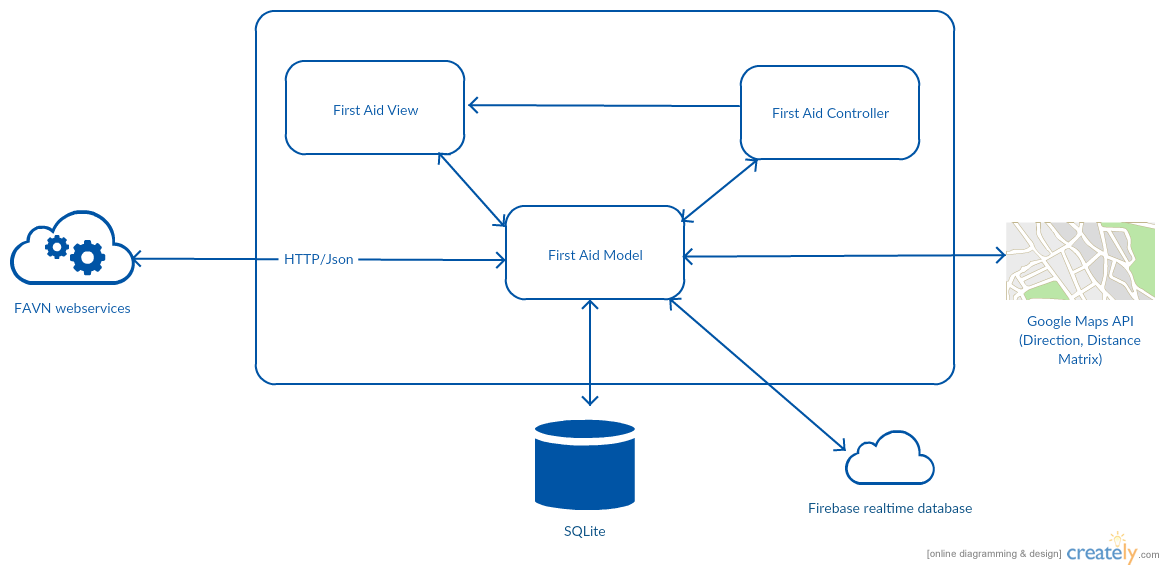


Figure -1: First Aid application architecture overview.

**Model** is data layer, responsible for managing the business logic and handling web service or database API.

**View** is UI layer, a visualization of the data from the Model.

**Controller** is logic layer, gets notified of the user’s behavior and updates the Model as needed.

## Ambulance Application

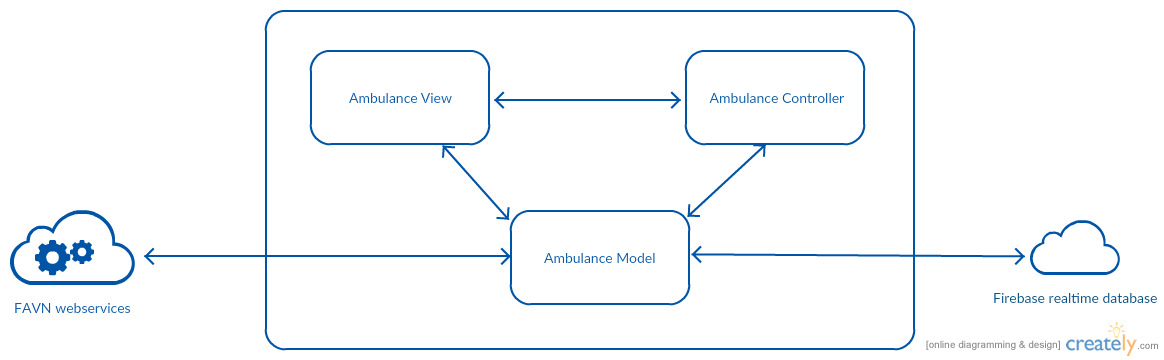


Figure 3-2: Ambulance application architecture overview.

As the First Aid Application's architecture, we also use MVC architecture for Ambulance application development.

## Dispatcher Application

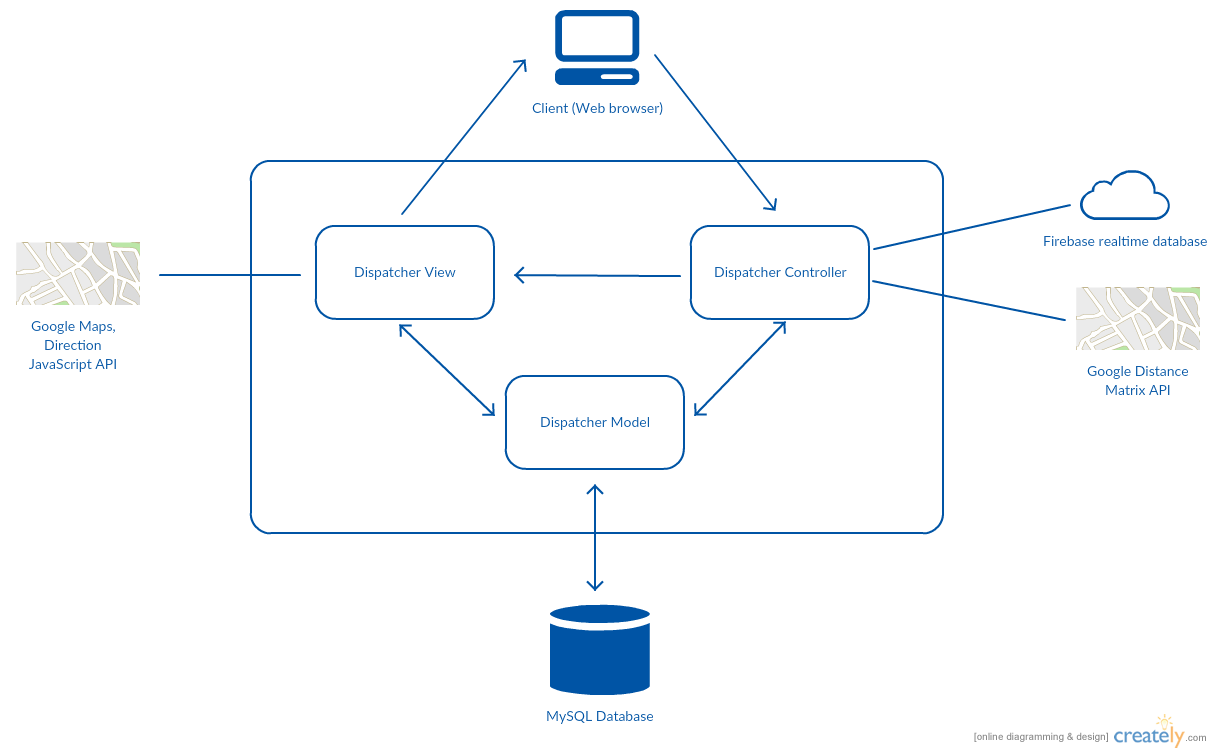


Figure 3-3: Dispatcher application architecture overview.

**Model** is where the application’s data objects are stored. A model object is in charge of encapsulating application state and one object could be related to other objects establishing a one-to-one or one-to-many relationship. The model object does not talk directly to a View, instead is made available to a controller, which accesses it when needed. When a model changes, typically it will notify its observers that a change has occurred.

**View** is what is presented to the users and how users interact with the system. The view is expected to render the model in a meaningful way to the user. In Dispatcher website, the view is made with .blade.php file including css, JavaScript or jQuery, it sends user gestures to controller and allows controller to select view.

**Controller** is the decision maker and the glue between the model and view; it handles user actions and gestures, and responds to user events. For example, in case a dispatcher needs to dispatch a car to a caller, when the dispatcher clicks “caller phone number” list item to create a new caller, the controller for that action is invoked. The controller will get an instant object of caller with data from database through caller model. The controller will then return view with that caller object to display, so that dispatcher can view the caller information.

## Admin/Expert Application

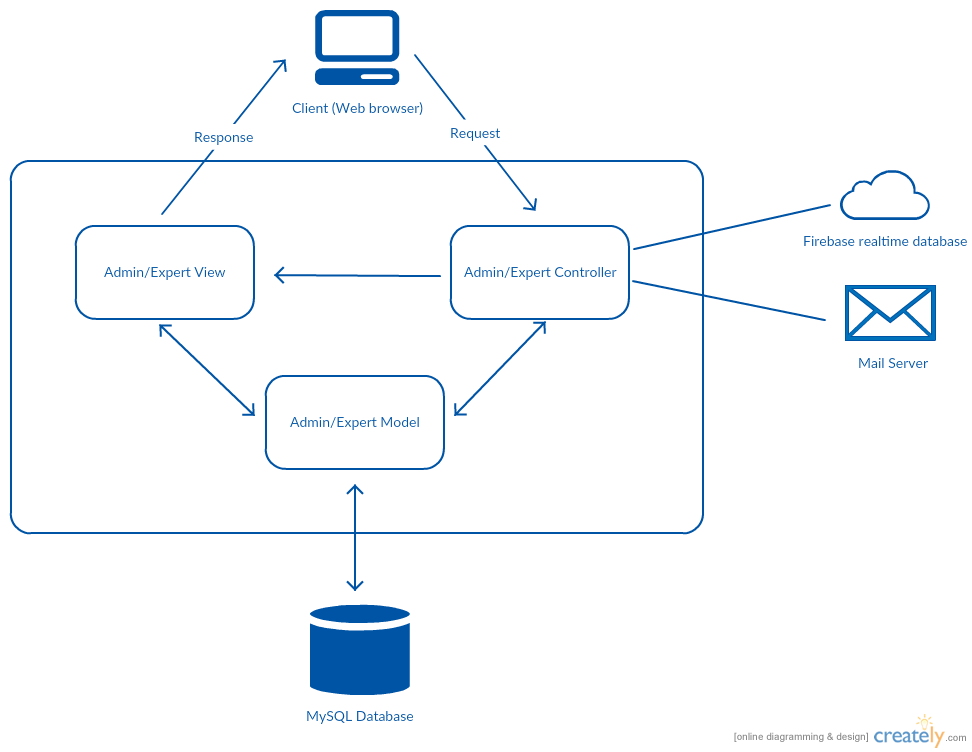


Figure 3-4: Admin/Expert website architecture overview.

As the Dispatcher Application's architecture, we also use MVC architecture for Admin/Expert application development.

# ARCHITECTURAL GOALS AND CONSTRAINTS

* **Availability:**
* According to common sense. We expect the system to be available 95% FAVN time. In case of necessity, users just install the application and setting up network to use the emergency services.
* **Maintainability:**
  + Coding standards and naming conventions:
    - Output of the project must include coding standards and naming conventions documentations. Implementation code must be easy to maintain.
    - All code must be clearly commented, including class, method documentations.
    - If some components are reused, the documentations of those components must also be included.
* Design:
  + - The design of the system must be loosely coupled that chances on some module will not affect others.
* Logging:
  + - All the errors should be logged, supporting for bug fixing and maintenance.
    - All strange or sensitive situations should also be logged.
* **Usability:**
  + Intuitiveness: all help/error messages are simple to understand; user can know exactly how to do each feature after one time using it.
* **Capacity and scalability:**
  + Throughput, storage and growth requirements.

# Use-case View

# Logical View

## Overview

Logical view is a logical way of view of elements in FAVN system as well as the relationships between them. It includes Package diagram and Class diagram. Package diagram describes the organization of packages and elements. Class Diagram provides an overview of the target system by describing the objects and classes inside the system and the relationships between them. It provides a wide variety of usages; from modeling the domain-specific data structure to detailed design of the target system.

* + Controller contain the interface between:
    - Associated models
    - Associated views
    - The input devices (e.g., keyboard, pointing device, time).
    - Send commands to the model to update the model's state.
  + Model is:
    - the domain-specific software simulation
    - Or implementation of the application's central structure.
  + View: deal with everything graphical:
    - Requests data from their model
    - Display the data.

## Architecturally Significant Design Packages

### First Aid Application

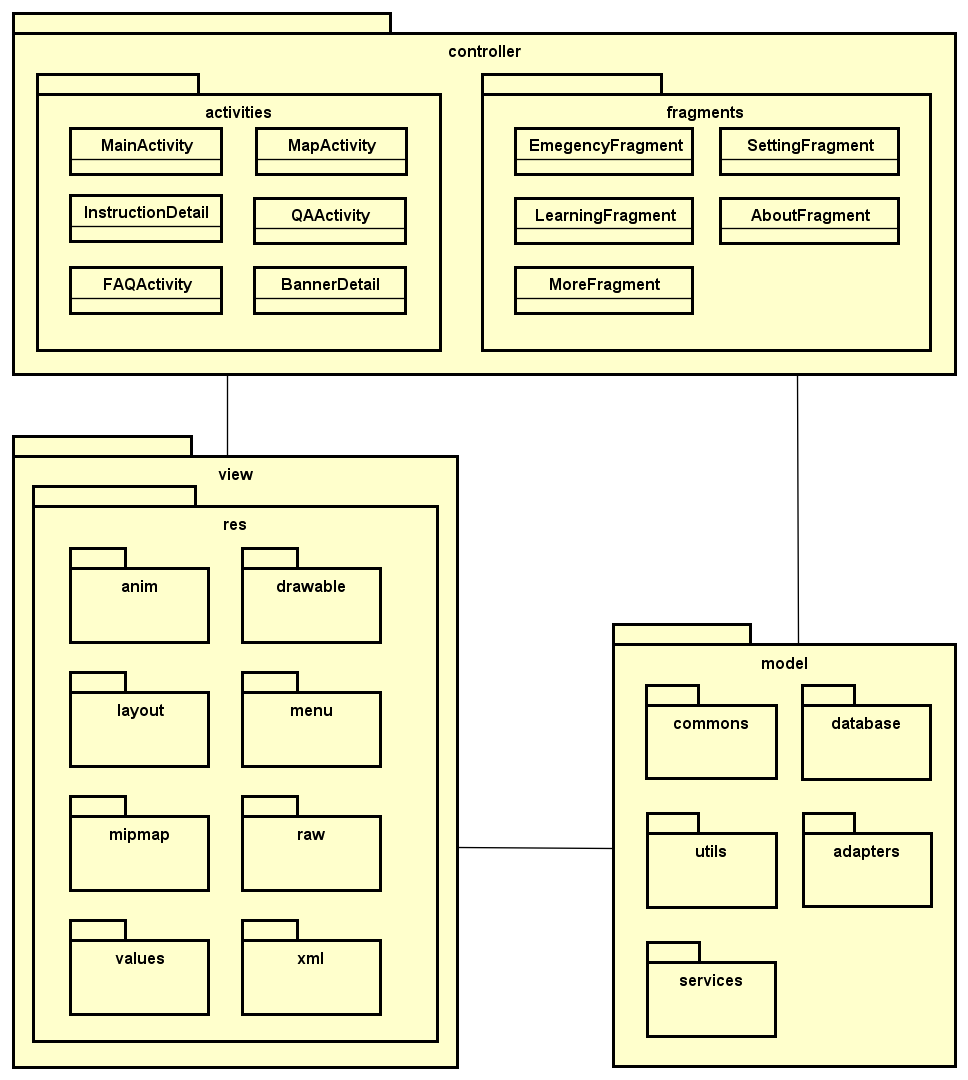


Figure 6-1: First Aid Application Package Diagram.

* **Model:**

|  |  |  |
| --- | --- | --- |
| No | Model class | Role |
|  | commons | Description entity of common’s class in sourcecode |
|  | databases | Description entity of database’s class in sourcecode |
|  | utils | Description entity of util’s class in sourcecode |
|  | adapters | Description entity of adapter’s class in sourcecode |
|  | services | Description entity of service’s class in sourcecode |

Table ‑: First Aid Application Model list

* **Controller:**

|  |  |  |
| --- | --- | --- |
| No | Controller class | Role |
|  | MainActivity | * Active Navigation bar |
|  | MapActivity | * Show |
|  | InstructionDetail |  |
|  | QAActivity |  |
|  | FAQActivity |  |
|  | CourseActivity |  |
|  | EmergencyFragment |  |
|  | LearningFragment |  |
|  | MoreFragment |  |
|  | SettingFragment |  |
|  | AboutFragment |  |

Table ‑: First Aid Application Controller list

* **View:**

Include many .xml file

### Ambulance Application

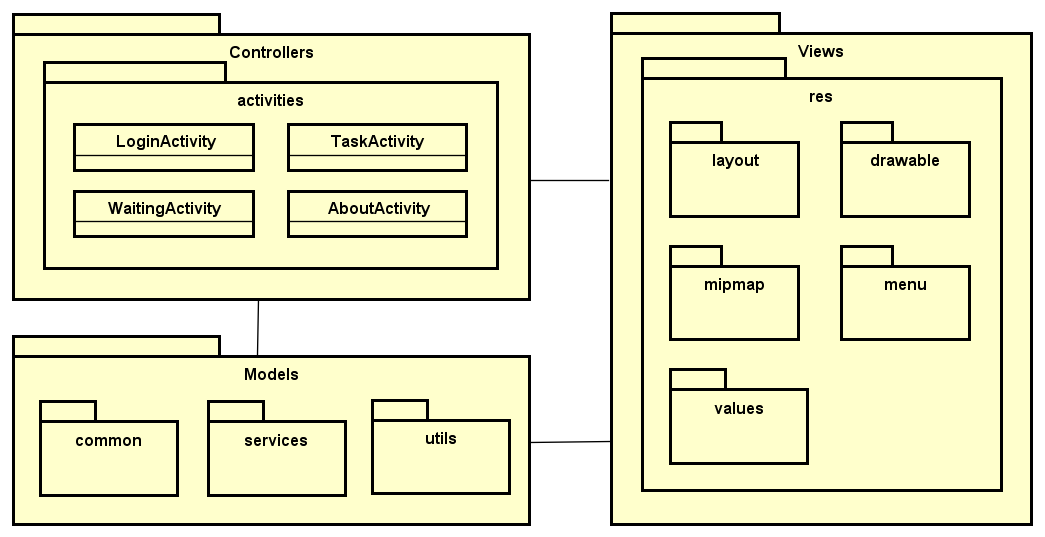


Figure 6-2: Ambulance Application Package Diagram.

* **Model:**

|  |  |  |
| --- | --- | --- |
| No | Model class | Role |
|  | common | Description entity of common’s class in sourcecode |
|  | services | Description entity of service’s class in sourcecode |
|  | utils | Description entity of util’s class in sourcecode |

Table ‑3: Ambulance Application Model list

* **Controller:**

|  |  |  |
| --- | --- | --- |
| No | Controller class | Role |
|  | LoginActivity |  |
|  | TaskActivity |  |
|  | WaittingActivity |  |
|  | AboutActivity |  |

Table ‑4: Ambulance Application Controller list

* **View:**

Include many .xml file

### Dispatcher Application

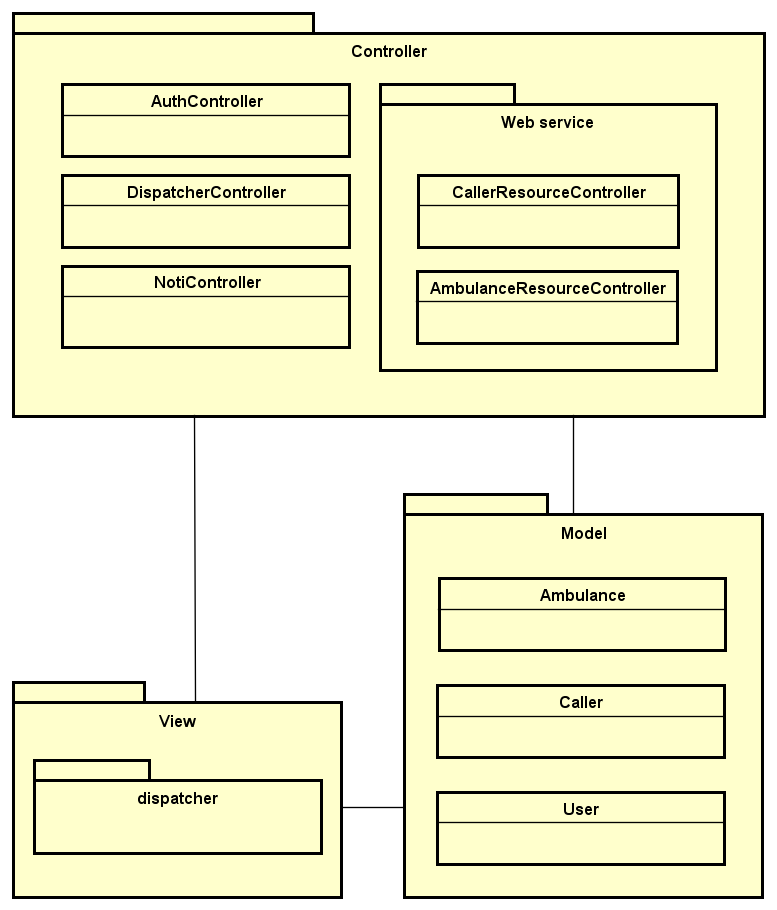


Figure 6-3: Dispatcher Application Package Diagram.

* **Model:**

|  |  |  |
| --- | --- | --- |
| No | Model class | Role |
|  | Ambulance | Description entity of Ambulance in database |
|  | Caller | Description entity of Caller in database |
|  | User | Description entity of User in database |

Table ‑5: Dispatcher Application Model list

* **Controller:**

|  |  |  |
| --- | --- | --- |
| No | Controller class | Role |
|  | AuthController | * Receive request login, logout from client. * Call method login, logout. * Respond login view and login, logout status. |
|  | DispatcherController | * Receive request to dispatch ambulance for a caller from client (dispatcher). * Call methods to handle match an ambulance with a caller. * Respond data back to Dispatcher view. |
|  | NotiController | * Receive request to push notification between ambulance, caller and dispatcher. * Hander request and return data. |
|  | CallerResourceController | RESTful controller that handles basic functionality for Caller table. (Create, read, update and delete records). |
|  | AmbulanceResourceController | RESTful controller that handles basic functionality for Ambulance table. (Create, read, update and delete records). |

Table ‑6: Dispatcher Application Controller list

* **View:**

Include many .blade.php file

### Admin/Expert Application

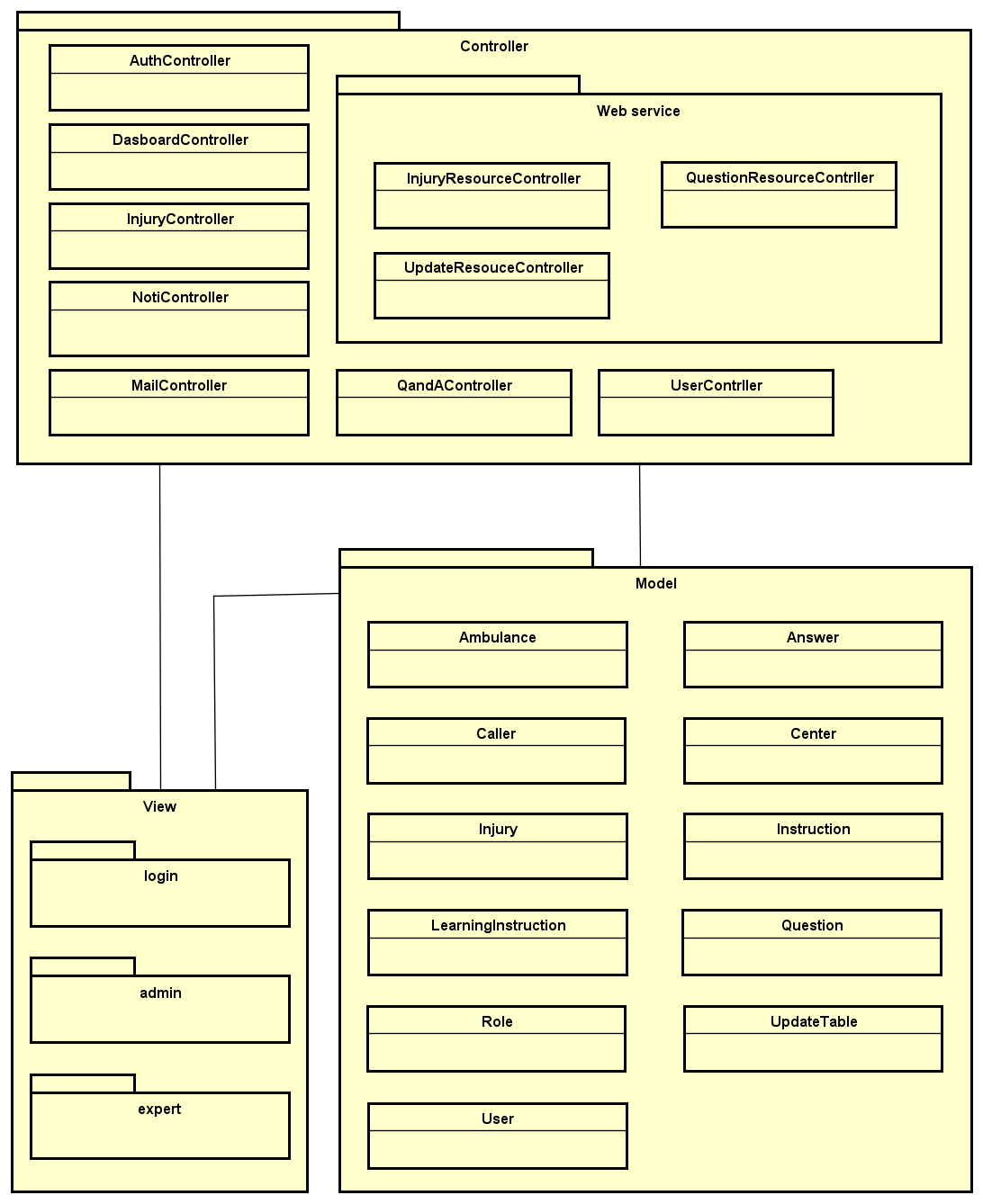


Figure 6-4: Admin/Expert Application Package Diagram.

* **Model:**

|  |  |  |
| --- | --- | --- |
| No | Model class | Role |
|  | Ambulance | Description entity of Ambulance in database |
|  | Answer | Description entity of Answer in database |
|  | Caller | Description entity of Caller in database |
|  | Center | Description entity of Center in database |
|  | Injury | Description entity of Injury in database |
|  | Instruction | Description entity of Instruction in database |
|  | LearningIntruction | Description entity of Learning Instruction in database |
|  | Question | Description entity of Question in database |
|  | Role | Description entity of Role in database |
|  | UpdateTable | Description entity of Update Table in database |
|  | User | Description entity of User in database |

Table ‑7: Admin/Expert Application Model list

* **Controller:**

|  |  |  |
| --- | --- | --- |
| No | Controller class | Role |
|  | AuthController | * Receive request login, logout from client. * Call method login, logout. * Respond login view and login, logout status. |
|  | DasboardController | * Receive request to show common information. * Call methods to handle. * Respond data back to view. |
|  | InjuryController | * Receive request to show, insert, delete, update, search injuries. * Hander request and return data. |
|  | NotiController | * Receive request to push notification between ambulance, caller and dispatcher. * Hander request and return data. |
|  | MailController | * Receive request to send mail from admin/expert to Fist Aid user. * Call methods to handle send mail. * Respond data back to view. |
|  | QandAController | * Receive request to show, reply or delete questions from Fist Aid user. * Call methods to handle.   Respond data back to view. |
|  | UserContrller | Receive request to show, insert, delete or update users. Call methods to handle. Respond data back to view. |
|  | InjuryResourceController | RESTful controller that handles basic functionality for Injury table. (Create, read, update and delete records). |
|  | UpdateResouceController | RESTful controller that handles basic functionality for Update table. (Create, read, update and delete records). |
|  | QuestionResourceContrller | RESTful controller that handles basic functionality for Question table. (Create, read, update and delete records). |

Table ‑8: Admin/Expert Application Controller list

* **View:**

Include many .blade.php file

# Process view

## Android app module process view

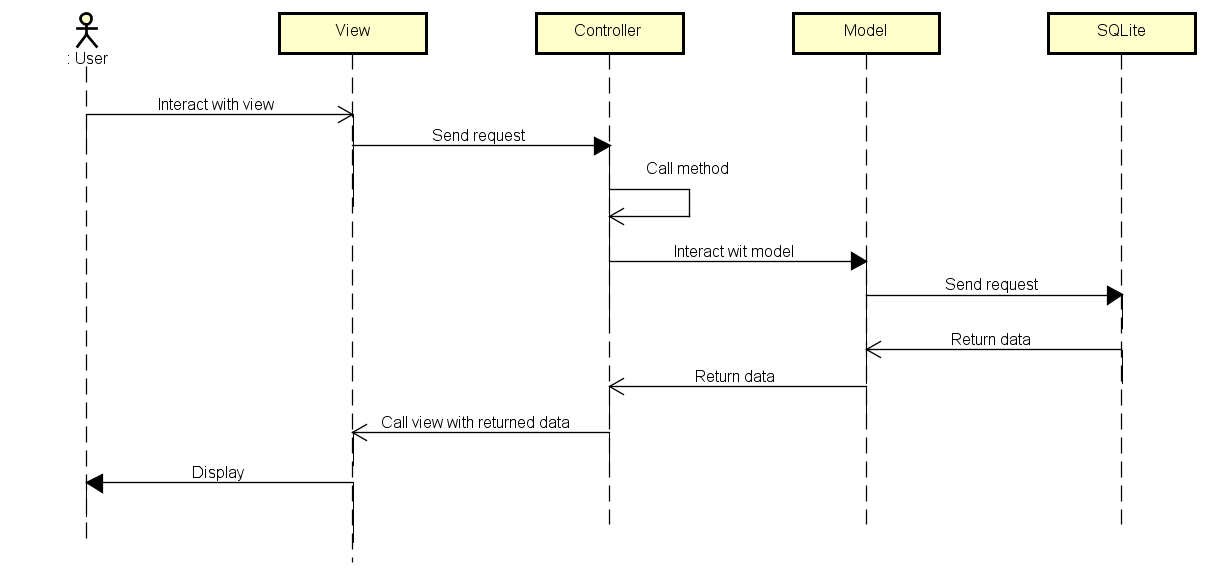


Figure 7-1: Android Module Overview Sequence Diagram.

## Web module process viewC:\Users\Hoang Gia\Desktop\Kienmt\capston\Diagrams\Process view web.png

Figure 7-2: Web Module Overview Sequence Diagram.

# Deployment View

## Android module deployment view

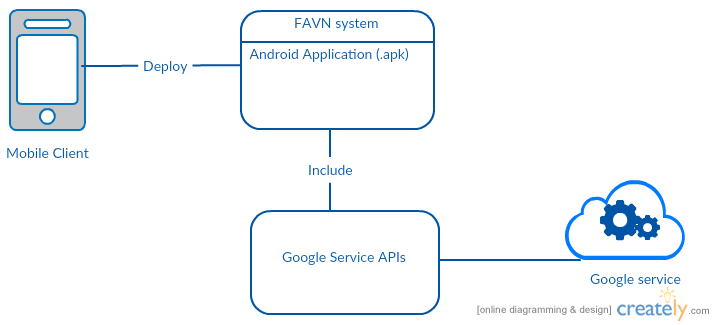


Figure ‑: Android Module Deployment Diagram.

|  |  |  |
| --- | --- | --- |
| **No** | **Name** | **Description** |
| 1 | Mobile Client | Client is android mobile to use system. |
| 3 | Google Service | Google Service APIs. |
| 4 | FAVN System | Install package (.apk file) |

**Table 8‑2:** Web Module Deployment Diagram Description.

## Web module deployment view

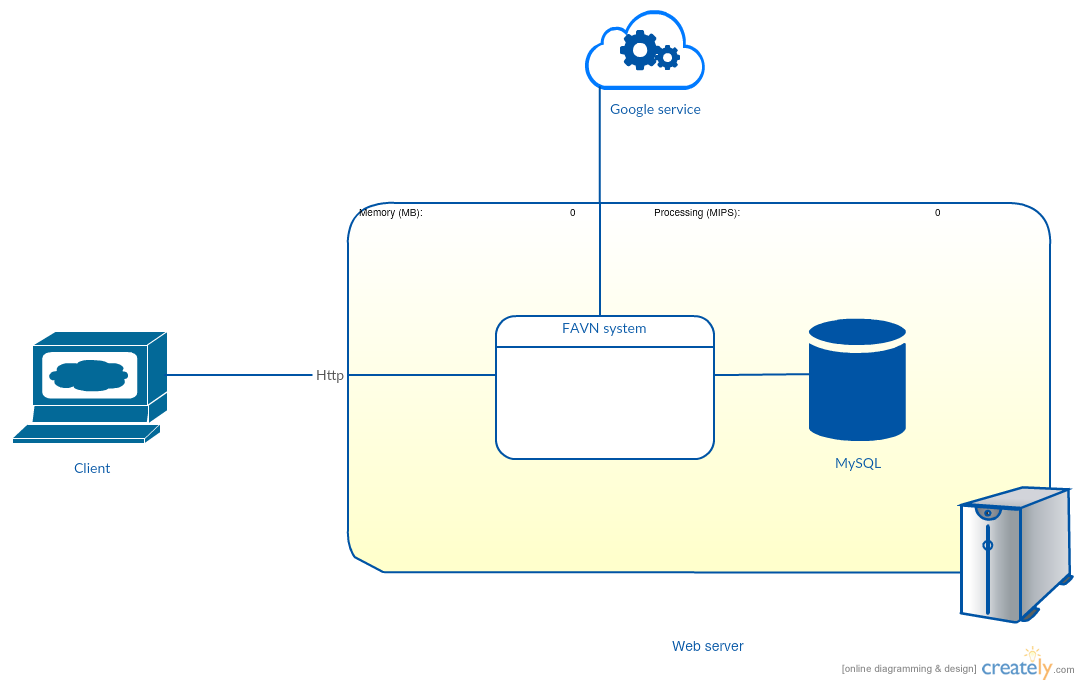


Figure ‑: Web Module Deployment Diagram.

|  |  |  |
| --- | --- | --- |
| **No** | **Name** | **Description** |
| 1 | Client | Client is web browser to use system. |
| 3 | Google Service | Google Service APIs. |
| 4 | MySQL | MySQL database use to store system’s data.  Using MySQL 5.7. |
| 5 | Web Server | Web server is hosted by Google Cloud Service. |

**Table 8‑2:** Web Module Deployment Diagram Description.

# Quality

Reference to: FAVN\_Software requirement specification\_v1.0\_EN.docx