Gudspot

Software Architecture Document

Version 1.0

Revision History

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 02/12/2021 | 1.0 | SAD using given template | Khôi |
| 18/12/2021 | 2.0 | Adding part 5 and part 6 | Khôi |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

2. Architectural Goals and Constraints 4

3. Use-Case Model 4

4. Logical View 4

4.1 Component: abc 4

5. Deployment 4

6. Implementation View 4

Software Architecture Document

# Introduction

**1.1 Purpose**

This document provides a comprehensive architectural overview of Gudspot, using a number of different architectural views to depict different aspects of Gudspot. It is intended to capture and convey the significant architectural decisions which have been made on Gudspot.

**1.2 Scope**

This Software Architecture Document provides an architectural overview of the C-Registration System. The C-Registration System is being developed by Wylie College to support online course registration.

**1.3 Overview**

This Software Architecture Document contains the following information:

Architectural Goals and Constraints - describes the software requirements and objectives that have some significant impact on the architecture.

Use-Case Model - describes the goals of the user, the interactions between the users and the system, and the required behavior of the system in satisfying these goals.

Logical View - describes the functionality that the system provides to end-users.

# Architectural Goals and Constraints

There are some key requirements and system constraints that have a significant bearing on the architecture. They are:

1/ Gudspot must be a lightweight website that is responsive and user-friendly.

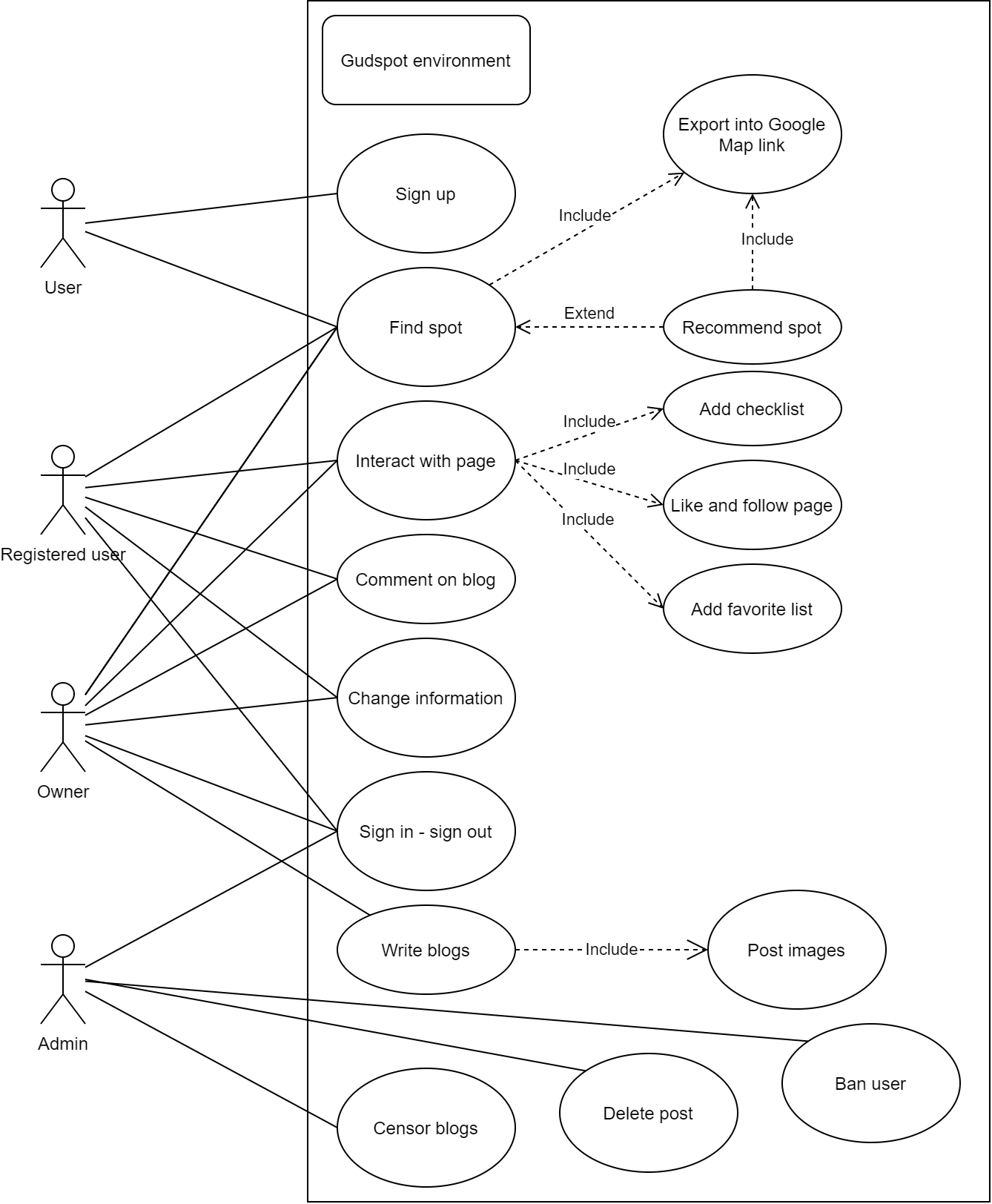
2/ All the user’s data must be secured and protected

3/ Stores on the platform must be wholesome, reputable and have good services

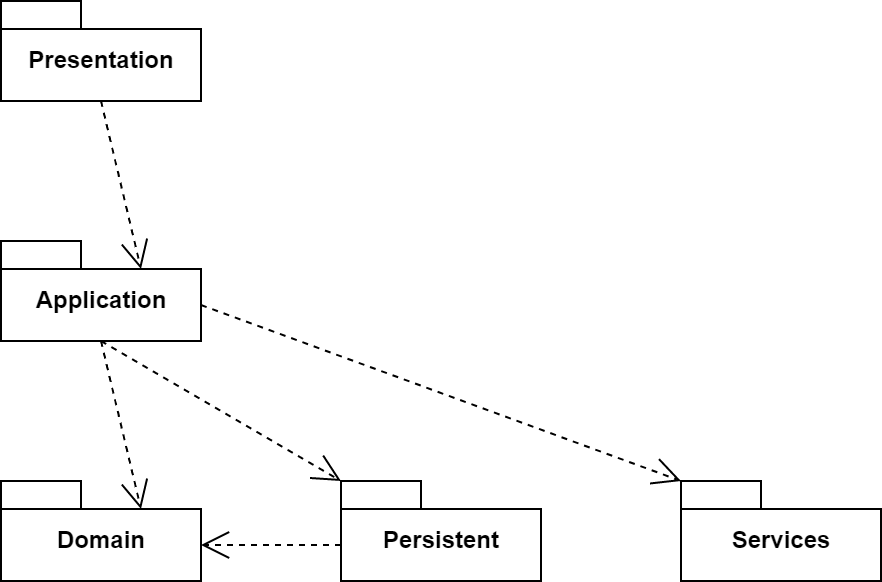
4/ Users that want to be the owner of a store on Gudspot must be carefully verified by the product team.

5/ The returned results from Gudspot search engine need to be diverse and appropriate for each time of year.

# Use-Case Model

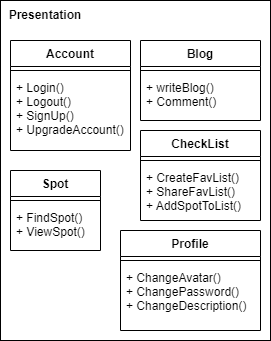
**

# Logical View

**

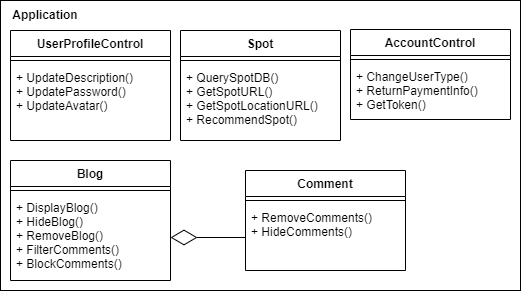
Hierarchical flow of packages: higher packages request lower package to provide necessary components.

## Component: Presentation

**

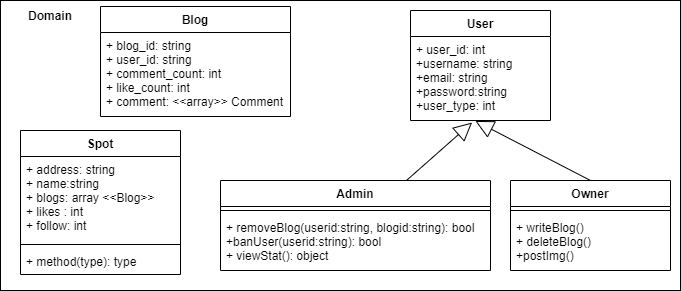
UI preface for user to interact with. These functions are interacted directly at the UI.

## Component: Application



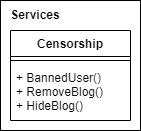
Functions is used within the system to control the workflows

## Component: Domain



Domain contains all the relevant data of the system for Application package use.

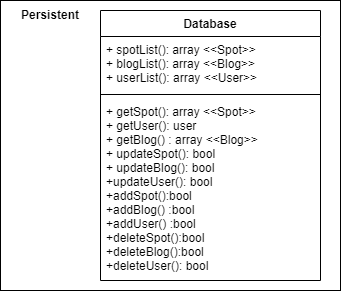
## Component: Service



Administrator functions to control the censorship: including users and blogs management.

## Component: Persistent

Database stores all the essential data, provided for the Domain package.



# Deployment

# 

5.1 Database Server

Contains all of the system data, retrieves data when a request comes.

5.2 Web Server

Information requested by client browser is stored here, then forward to db server.

5.3 Browser Device

Request, retrieve information from web server side to display on private monitor.

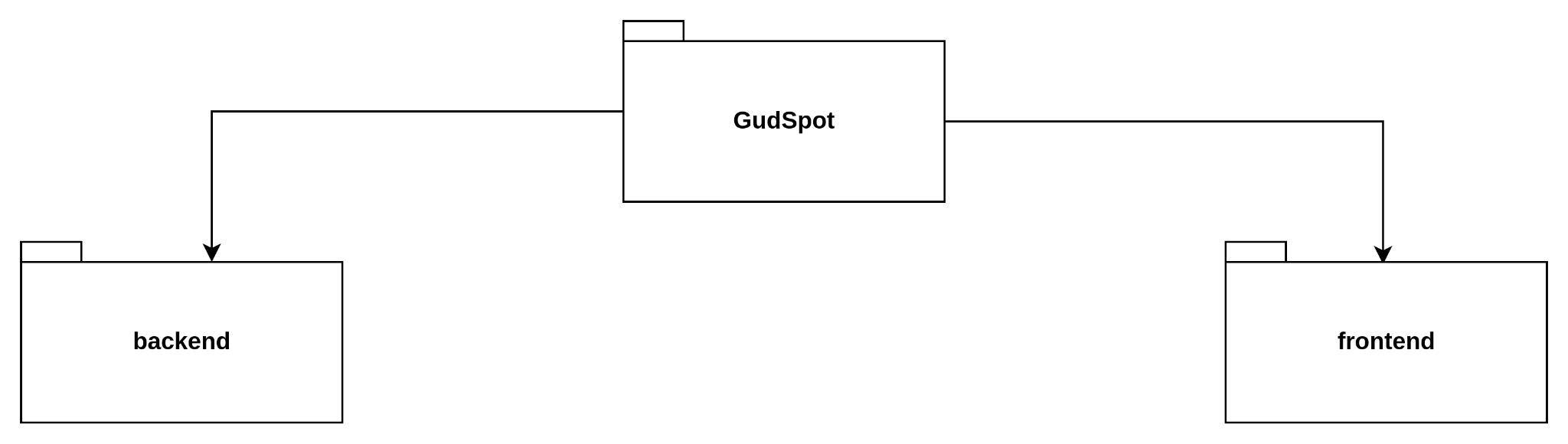
5.4 Private Monitor

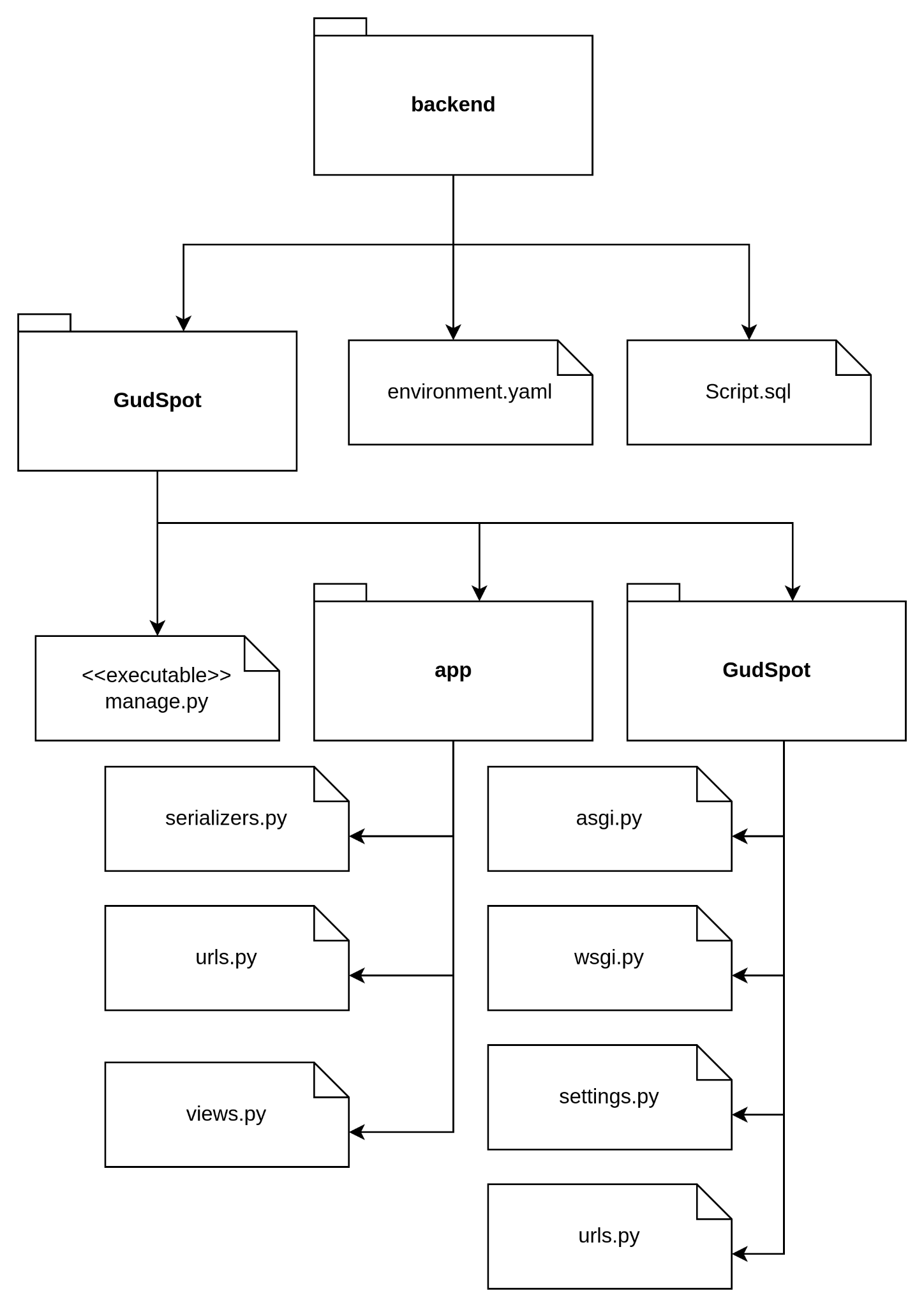
Information stored on web server is retrieved by browser , then displayed on private monitor

5.5 Private Processor

Supply necessities for the device to run the web.

# Implementation View

****



**Description: top-down description**

backend: contains files and folder of the project

GudSpot contains the project

environment.yaml: information about the developing environments, dependencies

Script.sql: additional tables construction in database

manage.py: main executable file that runs the whole server

app: contains files about the backend server

GudSpot: contains files to setup the whole project

serializers.py: contain classes that helps with serialization

urls.py: contains urls that supported by the app

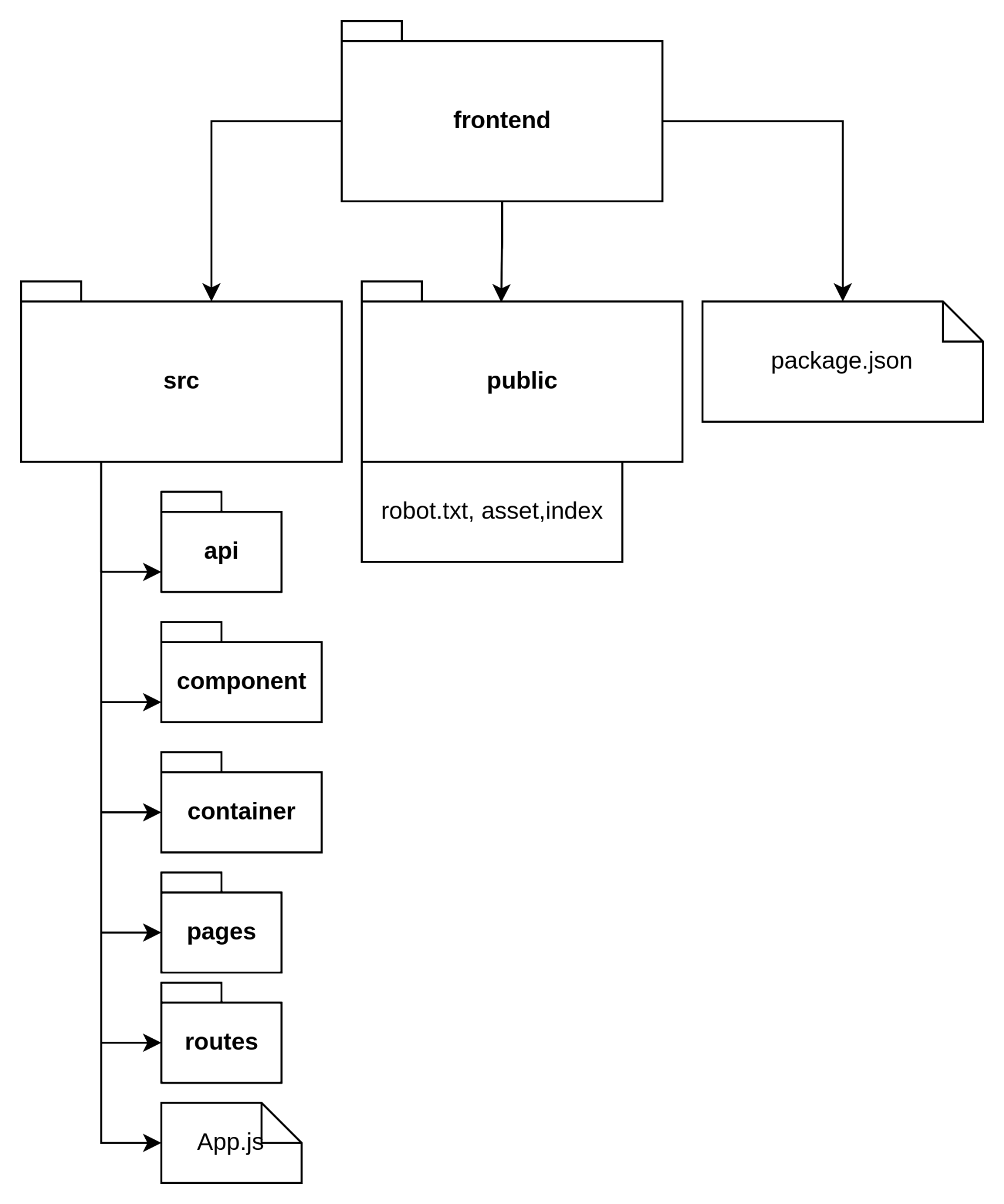
views.py: contain classes that communicate with other user via http requests

asgi.py: ASGI config for GudSpot project

wsgi.py: WSGI config for GudSpot project

urls.py: contains urls that supported by the whole project

settings.py: contains setting of the project, including dependencies, database setup

**

**Description: top-down description**

**frontend**: contains files and folder of the project

**src**: contains source of the react app

**public**: contains html file and asset folder including in image, logo

**package.json**: contains packages after installing (dependencies, devDependencies)

**api**: folder contains the api endpoint to the server

**component:** contains components relating to style and logic

**containers:** contains components including logics

**pages:** including pages

**routes:** the place to config routes of the app

**App.js:** the child of the index.js file

**index.js:** parent file including another component to build