VIDHYADEEP INSTITUTE OF COMPUTER & INFORMATION TECHNOLOGY, ANITA-KIM.

VEER NARMAD SOUTH GUJARAT UNIVERSITY (VNSGU)

PROJECT REPORT ON

“**DG-Estate**”

as a partial requirement for the degree of

bachelor of computer applications

[b.c.a]

3 YEARS integrated cource

YEAR: 2019 – 2022

SUBMITTED BY:

Darshan Nariya

(E19110403000110036)

GUIDED BY:

Drashti Bhatt

(Internal Guide)

ORGANIZATION:

KG Krunch

Solutions

VIDHYADEEP INSTITUTE OF COMPUTER & INFORMATION TECHNOLOGY, ANITA-KIM.

VEER NARMAD SOUTH GUJARAT UNIVERSITY (VNSGU)

PROJECT REPORT ON

“**DG-Estate**”

as a partial requirement for the degree of

bachelor of computer applications

[b.c.a]

3 YEARS integrated cource

YEAR: 2019 – 2022

SUBMITTED BY:

Darshan Nariya

(E19110403000110036)

GUIDED BY:

Drashti Bhatt

(Internal Guide)

ORGANIZATION:

KG Krunch

Solutions

# Acknowledgement

I

n successfully completing of this project, many websites, blogs, posts, videos, discussions have helped me.[[1]](#footnote-1) I would like to thank all these, I have also upvoted these blogs, videos whenever it was possible.

Every single problem in this project gave me a challenge, and I like challenges, it gave me huge confidence boost whenever I solved any challenging problem.

The Company where I worked (KG Krunch Solutions) was also a big help, while I didn’t make my project there, but I learned a lot while working there.[[2]](#footnote-2) I will be always thankful to the company and team.

This project doesn’t just show how much I know it also shows how much I don’t know. So, I’m looking forward to learn new things every day.

Now let’s come to the college “VIDHYADEEP INSTITUTE OF COMPUTER & INFORMATION TECHNOLOGY”, while admin department was a pain to work with, where I had some uneasy experiences, our department’s professors were easy on us, gave us lot of freedom. It was an experience in its own, and I thank all the faculty. [[3]](#footnote-3)

For upcoming days and new things, I’m ready, one day at a time, finally thanks to all the people and things that helped me directly or indirectly.

Darshan Nariya

# Index

[Acknowledgement I](#_Toc99358031)

[Index II](#_Toc99358032)

[1 Introduction Profiles 1](#_Toc99358033)

[1.1 Project Profile 1](#_Toc99358034)

[1.2 Company Profile 2](#_Toc99358035)

[2 Objective of Project 3](#_Toc99358036)

[2.1 User 3](#_Toc99358037)

[2.2 Admin 3](#_Toc99358038)

[2.3 Root 3](#_Toc99358039)

[3 Environment Description 4](#_Toc99358040)

[3.1 Hardware Requirements 4](#_Toc99358041)

[3.1.1 Server 4](#_Toc99358042)

[3.1.2 Client 4](#_Toc99358043)

[3.2 Software Requirements 5](#_Toc99358044)

[3.2.1 Server 5](#_Toc99358045)

[3.2.2 Client 5](#_Toc99358046)

[3.3 Development Environment 6](#_Toc99358047)

[3.3.1 Hardware 6](#_Toc99358048)

[3.3.2 Software 6](#_Toc99358049)

[3.3.3 Dependencies 7](#_Toc99358050)

[3.4 Methods & Technologies 8](#_Toc99358051)

[3.4.1 PHP 8](#_Toc99358052)

[3.4.2 MySQL 8](#_Toc99358053)

[3.4.3 Laravel 9](#_Toc99358054)

[3.4.4 MVC 9](#_Toc99358055)

[3.4.5 Bootstrap 11](#_Toc99358056)

[3.4.6 JQuery 11](#_Toc99358057)

[3.4.7 Composer 11](#_Toc99358058)

[4 System Planning 12](#_Toc99358059)

[4.1 Feasibility Study 12](#_Toc99358060)

[4.1.1 Summary 12](#_Toc99358061)

[4.1.2 Technological Considerations 13](#_Toc99358062)

[4.2 Requirement Specification 14](#_Toc99358063)

[4.2.1 Users 14](#_Toc99358064)

[4.2.2 Admin 15](#_Toc99358065)

[4.2.3 Root 16](#_Toc99358066)

[5 Proposed System 17](#_Toc99358067)

[5.1 Summary 17](#_Toc99358068)

[5.2 Scope 17](#_Toc99358069)

[6 Detail Planning 18](#_Toc99358070)

[6.1 Data-Flow Diagrams 18](#_Toc99358071)

[6.1.1 Context Level DFD 20](#_Toc99358072)

[6.1.2 Admin DFDs 21](#_Toc99358073)

[6.1.3 Root DFDs 29](#_Toc99358074)

[6.1.4 User DFDs 37](#_Toc99358075)

[6.2 Database Structure 41](#_Toc99358076)

[6.3 Entity Relation Diagram 46](#_Toc99358077)

[7 Design Report 49](#_Toc99358078)

[8 Testing Reports 50](#_Toc99358079)

[9 Limitations of the System 51](#_Toc99358080)

[10 Future Enhancement of the project 52](#_Toc99358081)

[11 References 53](#_Toc99358082)

# Introduction Profiles

## Project Profile

|  |  |
| --- | --- |
| Title | DG-Estate |
| Organization | KG-Krunch Solutions |
| Category | Local Property listing Website |
| Duration | 3 Months |
| Front-End | Blade (Laravel Framework) |
| Back-End | Laravel Framework (PHP) + MySQL |
| Guide | Drashti Bhatt (Internal Guide) |
| Submitted by | Darshan Arvind Bhai Nariya  (E19110403000110036) |
| Submitted to | Vidhyadeep Institute of Computer & Information Technology |

## Company Profile

|  |  |
| --- | --- |
| Company Name | KG Krunch Solutions |
| Proprietor Name | Ashish Gajera |
| Business Type | Solutions Provider |
| Address |  |
| Contact No |  |
| Website |  |
| Email |  |
| Service Range |  |

# Objective of Project

## User

The main User side Objectives are:

* To enable users to surf all listed properties.
* Users can surf properties without the need of creating account.
* Users can filter properties by category, city, and search by property title.
* Users can create account and manage their profile, save properties, review properties, and change their password.

## Admin

The main Admin side Objectives are:

* Admin can add, and update existing category, facilities, city, property.
* Admin can add and delete images in property gallery.
* Admin can delete user reviews.
* Admin can give other users admin access.

## Root

The main Root side Objectives are:

* Root has access to all above features.
* Root is only one who can delete existing category, facility, city, property, and users.
* Root can manage website’s Home, About, FAQ, Terms page in CMS.
* Root can manage website’s other settings in Site Settings option.

# Environment Description

## Hardware Requirements

### Server

|  |  |
| --- | --- |
| Processor | 3rd Gen and above |
| Memory | 4 GB and above |
| Storage | 6 GB and above |

### Client

|  |  |
| --- | --- |
| Processor | 3rd Gen and above |
| Memory | 4 GB and above |
| Storage | 6 GB and above |

## Software Requirements

### Server

|  |  |
| --- | --- |
| Operating System | Windows 7 or above (10 recommended),  any Linux based, 64bit arch. |
| Server | Apache v 2.4 and above |
| Php | Php v 6 and above |
| MySQL | MySQL v 7 and above |

### Client

|  |  |
| --- | --- |
| Operating System | Windows 7 or above (10 recommended), 64bit arch. |
| Browser | Any Chromium based Browsers |

## Development Environment

### Hardware

|  |  |
| --- | --- |
| Processor | Ryzen 5 3550H |
| Memory | 8 GB |
| Storage | 256 GB |

### Software

|  |  |
| --- | --- |
| Windows | 10/11 64bit |
| WAMP server | V 4.1 |
| Chrome | V 99.0.4844.82 64bit |
| Git | Windows V 2.34.1 |
| Composer | V 2.2.3 |
| VS Code | V 1.50^ |
| Diagrams.net/Draw.io | V 17.2.3 |

### Dependencies

|  |  |
| --- | --- |
| Laravel | V 8.x |
| Bootstrap | V 5.1.3 |
| Bootstrap Examples | V 5.1 |
| JQuery | V 3.6.0 |
| Font Awesome | V 4 & V 5 |
| Data Tables | V 1.11.4 |
| Data Tables Bootstrap | V 1.11.4 |
| Fancy Apps | V 4 Fancy Box & Carousal |
| CKEditor | V 4 |

## Methods & Technologies

### PHP

PHP is a general-purpose scripting language geared toward web development. It was originally created by Danish-Canadian programmer Rasmus Lerdorf in 1994.The PHP reference implementation is now produced by The PHP Group. PHP originally stood for Personal Home Page, but it now stands for the recursive initialism PHP: Hypertext Pre-processor.

PHP code is usually processed on a web server by a PHP interpreter implemented as a module, a daemon or as a Common Gateway Interface (CGI) executable. On a web server, the result of the interpreted and executed PHP code which may be any type of data, such as generated HTML or binary image data would form the whole or part of an HTTP response. Various web template systems, web content management systems, and web frameworks exist which can be employed to orchestrate or facilitate the generation of that response.

### MySQL

MySQL is an open-source relational database management system (RDBMS). Its name is combinations of “My”, the name of co-founder Michael Widenius' daughter, and "SQL", the abbreviation for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. For proprietary use, several paid editions are available, and offer additional functionality.

MySQL is a central component of the LAMP open-source web application software stack (and other "AMP" stacks). LAMP is an acronym for "Linux, Apache, MySQL, and Perl/PHP/Python". Applications that use the MySQL database include: TYPO3, Mode, Joomla, WordPress, php DB, MyDB, and Dural. MySQL is also used in many high-profile, large-scale websites, including Google (though not for searches), Facebook, Twitter, Flicker, and YouTube.

### Laravel

Laravel is a free, open-source PHP web framework, created by “Taylor Otwell” and intended for the development of web applications following the model-view-controller (MVC) architectural pattern and based on Symfony. Some of the features of Laravel are a modular packaging system with a dedicated dependency manager, different ways for accessing relational databases, utilities that aid in application deployment and maintenance, and its orientation toward syntactic sugar.

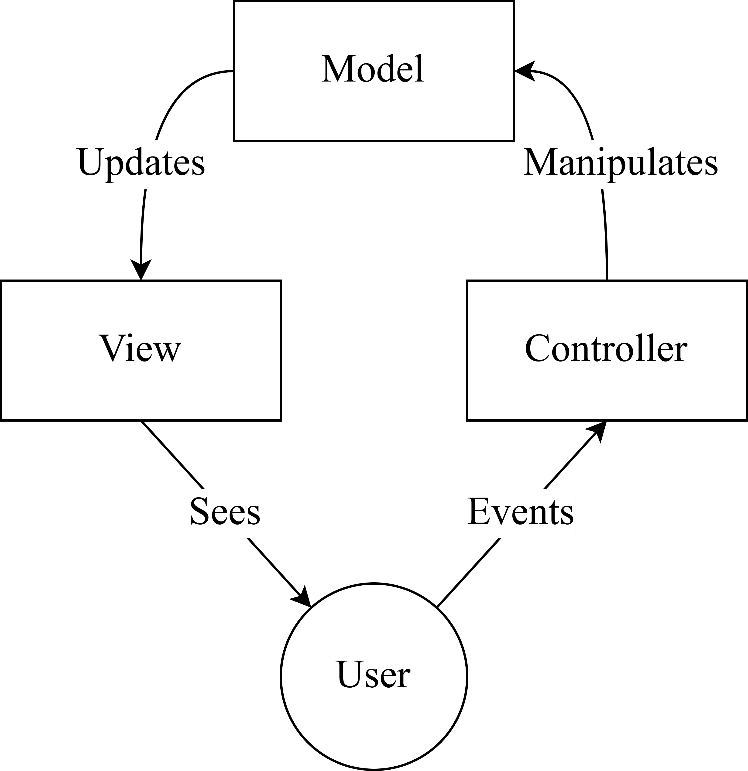
### MVC

Model-view-controller (MVC) is a software design pattern commonly used for developing user interfaces that divide the related program logic into three interconnected elements. This is done to separate internal representations of information from the ways information is presented to and accepted from the user.

Traditionally used for desktop graphical user interfaces (GUIs), this pattern became popular for designing web applications. Popular programming languages have MVC frameworks that facilitate implementation of the pattern.

**The Model** component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

**The View** component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.



**Controllers** act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.

### Bootstrap

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains HTML, CSS and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

### JQuery

jQuery is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animation, and Ajax. It is free, open-source software using the permissive MIT License. As of May 2019, jQuery is used by 73% of the 10 million most popular websites. Web analysis indicates that it is the most widely deployed JavaScript library by a large margin, having at least 3 to 4 times more usage than any other JavaScript library.

**jQuery'**s syntax is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications. jQuery also provides capabilities for developers to create plug-ins on top of the JavaScript library. This enables developers to create abstractions for low-level interaction and animation, advanced effects and high-level, theme-able widgets. The modular approach to the jQuery library allows the creation of powerful dynamic web pages and Web applications.

### Composer

Composer is an application-level package manager for the PHP programming language that provides a standard format for managing dependencies of PHP software and required libraries. It was developed by Nils Adermann and Jordi Boggiano, who continue to manage the project. They began development in April 2011 and first released it on March 1, 2012. Composer is strongly inspired by Node.js's “npm” and Ruby's “bundler”. The project's dependency solving algorithm started out as a PHP-based port of openSUSE's libzypp satsolver.

# System Planning

## Feasibility Study

### Summary

DG-Estate is property listing website where properties can only be listed by admin and root, while common users can surf through verity of properties by category, city, and search them by their title, logged in users can save properties, give reviews and manage their profile.

Real Estate website has so much potential to grow up in future, website can be upgraded into a platform where common users can also post their property for rent/sale, but these are future possibilities so we have to make this project scalable for the future.

For future scalability we can opt for Frameworks of any particular language, MVC architecture is preferable for such objectives mentioned above.

### Technological Considerations

For these objectives described in summary we have to handle a database for that and all these handling have to be done on server side for security purpose. below are few server-side languages can be considered.

**Server-Side Languages**:

* Node.js (JavaScript)
* PHP
* Java
* Ruby
* Python

From above list **PHP** is great option because it can be embedded inside HTML syntax, and has many frameworks which makes many tasks easy.

**PHP Frameworks**:

* Laravel
* CodeIgniter
* Symfony

Above three are most popular framework of PHP and using one of these will also make changes in website later easier, because they use MVC architecture.

**Laravel** is huge consideration because of its eloquent modelling method which makes database queries more human readable, also its syntax is convenient once you get hang of it.

## Requirement Specification

Requirements of client describe as follow:

* System should be easy to operate.
* System should provide secure and accurate data.

### Users

**Users** are registered users whom have signed up in website. New Users have default type ‘U’.

Login

* Registered users have to insert their own registered email and password in login form to successfully log in.
* If User’s entered email is not registered show error that email is not registered.
* If User’s entered password doesn’t match then show invalid credentials error message.
* If credentials match add user in session.

Signup

* To Signup User has to provide name, email, and password with confirmation.
* If entered email already exists then show error.
* If password doesn’t match with confirmation show error.
* If no errors occurs then register user.

Logout

* Check if user is logged in first.
* If user is logged in then remove user from session.

User Profile

* Login required.
* Profile includes image, bio, email options.

User Change Password

* Login required.
* User have to provide old password and new password with confirmation to change password.

Save Property

* Login required.
* If user is logged in show save property button in properties.
* Saved properties can be seen in specific saved page.

Review

* Login required.
* Logged in users can give reviews to properties.

### Admin

User who is type ‘A’ is considered **Admin**.

Login

* Admin have to login in admin specific page to access Admi Panel.
* After entering correct username and password admin session is started.
* Admin is redirected to Admi Panel dashboard after successful login.

Logout

* Login required.
* If admin is logged in remove admin session.

Category

* Login required.
* Admin can Add/Update Categories.

City

* Login required.
* Admin can Add/Update Cities.

Facility

* Login required.
* Admin can Add/Update Facilities.

Property

* Login required.
* Admin can Add/Update Properties.

Gallery

* Login required.
* Admin can Add/Delete Images in property Gallery.

Reviews

* Login required.
* Admin can Delete user’s reviews.

Users

* Login required.
* Admin can give Admin access to other users.

### Root

**Root** is super user of this website who has access to all the features and functionalities available on the website. All functionalities mentioned for Users and Admin is also accessible by Root. User with type ‘R’ is considered Root.

* In addition of above accessibilities Root has few more options to manage
* Root can Delete Category/City/Facility/Property/User.
* Root can modify Page content from CMS menu which is only accessible by type ‘R’ User or Root user.
* Root can change site title, logo, and contact details from Site Settings option.

# Proposed System

## Summary

“DG-Estate” enables Users to surf local properties listed by Admin/Root by Category, City and search them by property title. Users can then contact individual property dealer who listed his property by the contact information they provided.

Users can save , review properties if they are logged in.

Website allows management by two levels Admin and Root. Admin and Root has most Access and Functionalities in this website, and Root has more Control than Admin.

## Scope

For now, Scope of this website “DG-Estate” is Local, but it will be accessible to the global if client chooses to select global hosting.

Admin Panel allows Admin and Root to manage website content. Admin Panel allows access according to the type of user that is logged in.

Website is using MVC structure which will help in future scalability if in future client thinks about adding new features.

# Detail Planning

## Data-Flow Diagrams

A Data-Flow Diagram is a way of representing a flow of data through a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow — there are no decision rules and no loops.

Symbols and Notations Used in DFDs

1. External entity: an outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.
2. Process: any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules. A short label is used to describe the process, such as “Submit payment.”
3. Data store: files or repositories that hold information for later use, such as a database table or a membership form. Each data store receives a simple label, such as “Orders.”
4. Data flow: the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labelled with a short data name, like “Billing details.”

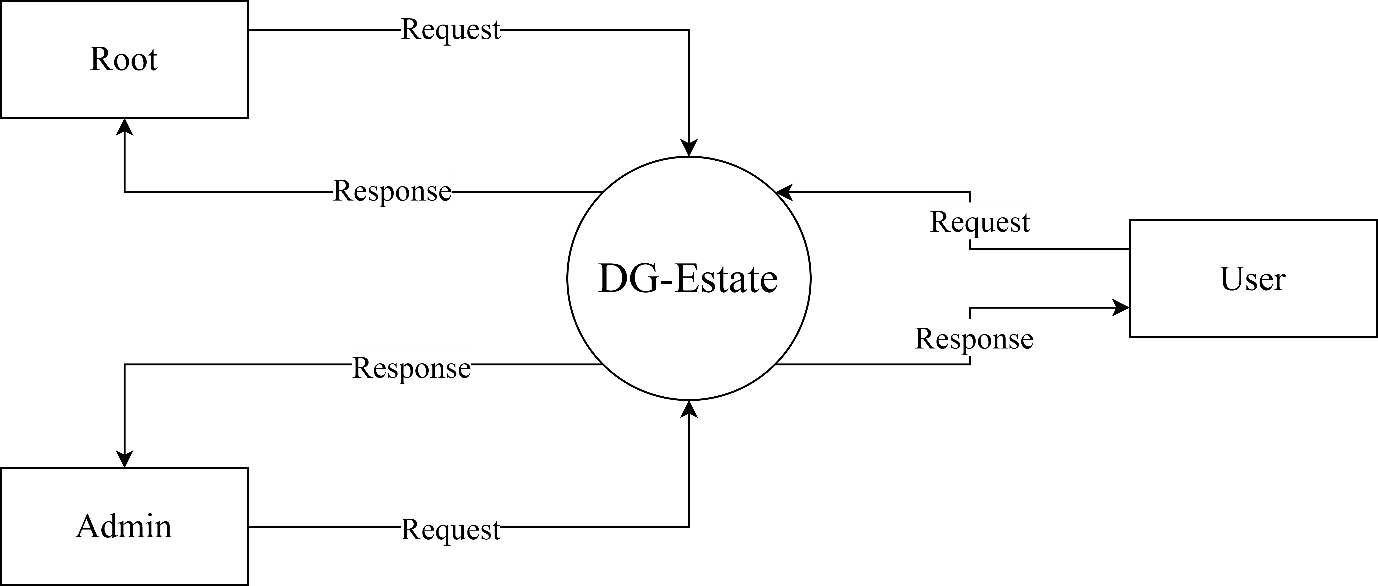
|  |  |
| --- | --- |
| Notation | Symbol |
| External Entity |  |
| Process |  |
| Data Store |  |
| Data Flow |  |

Types of Data-Flow Diagram

A data flow diagram can dive into progressively more detail by using levels and layers, zeroing in on a particular piece. DFD levels are numbered 0, 1 or 2, and occasionally go to even Level 3 or beyond.

* DFD Level 0 is also called a Context Diagram. It’s a basic overview of the whole system or process being analysed or modelled. It’s designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities.
* DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram. You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its subprocesses.
* DFD Level 2 then goes one step deeper into parts of Level 1. It may require more text to reach the necessary level of detail about the system’s functioning.

### Context Level DFD

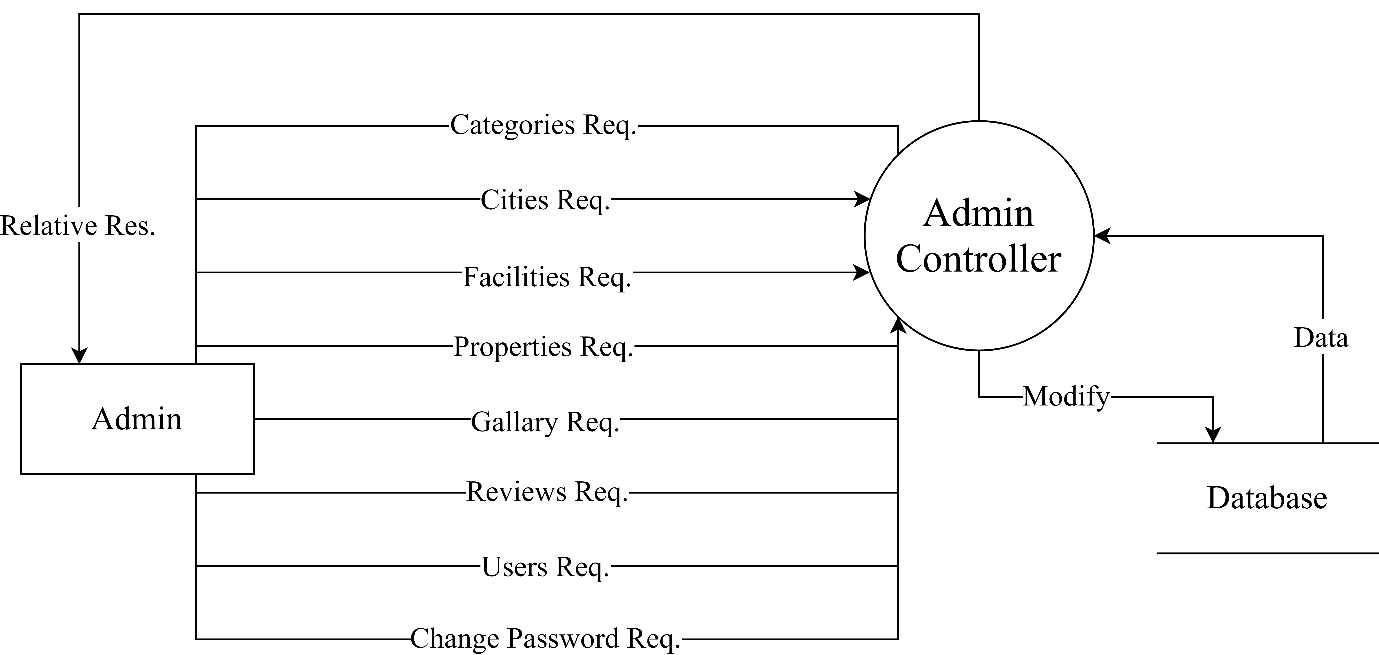


### Admin DFDs

Admin level 0



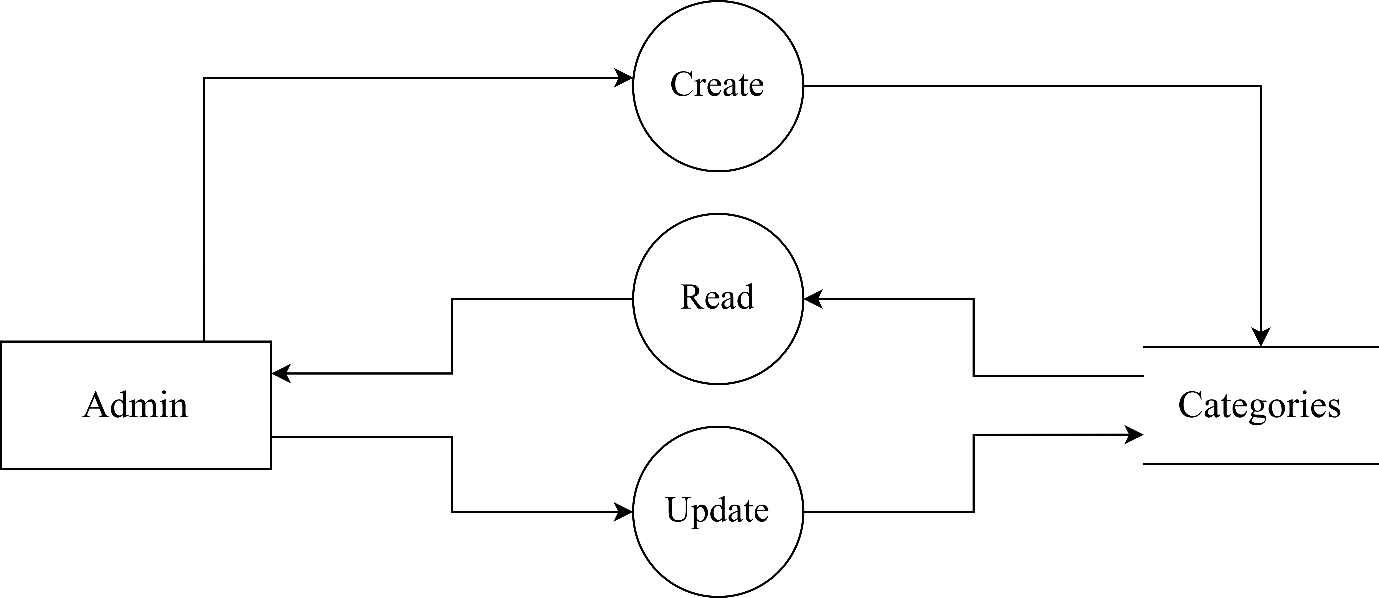
Admin level 1



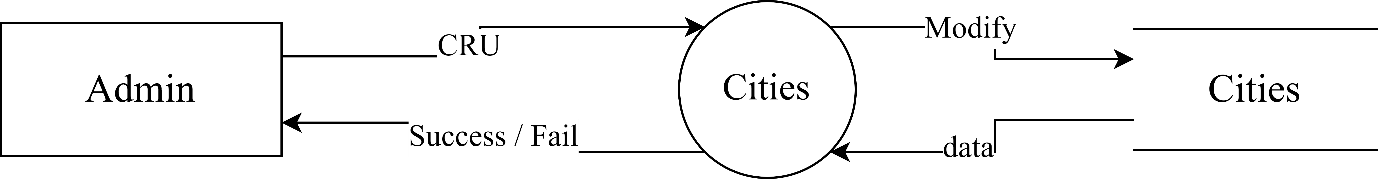
Category level 0



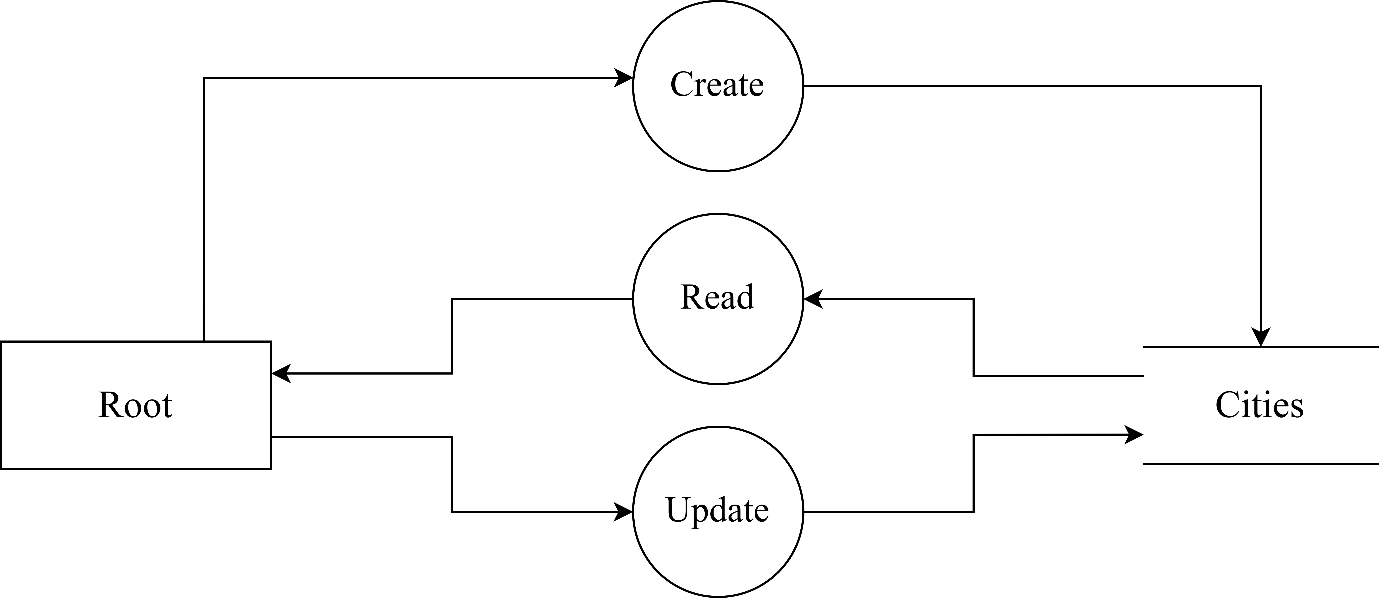
Category level 1



City level 0



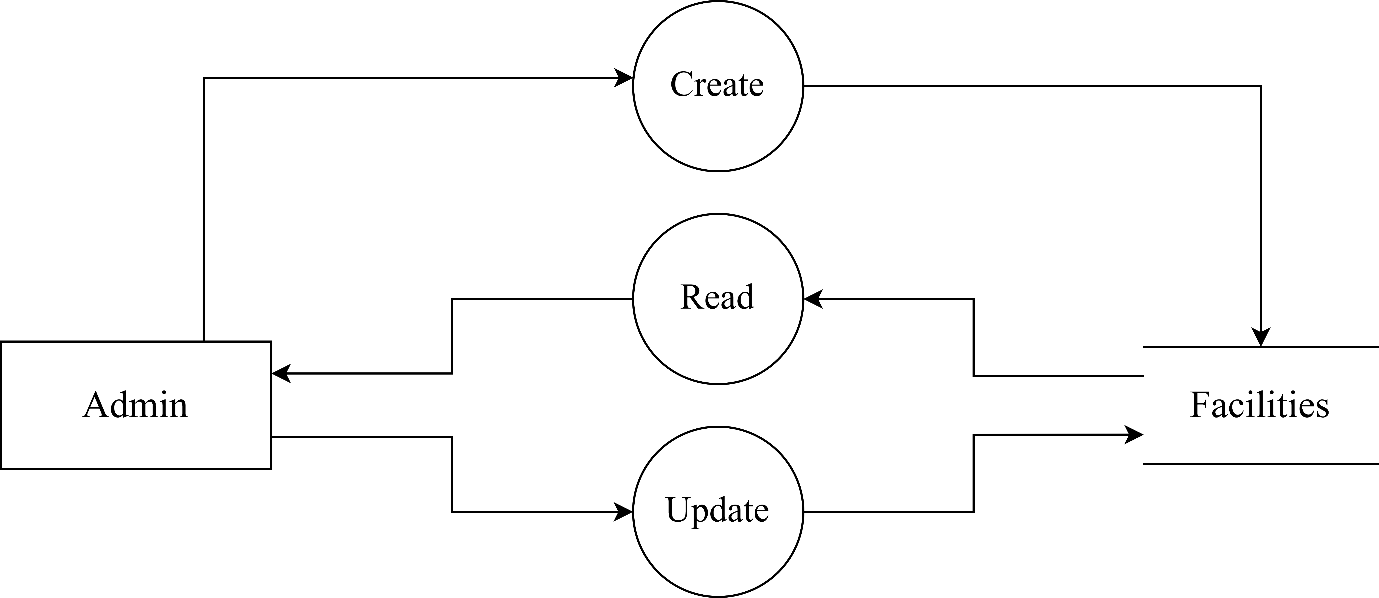
City level 1



Facility level 0



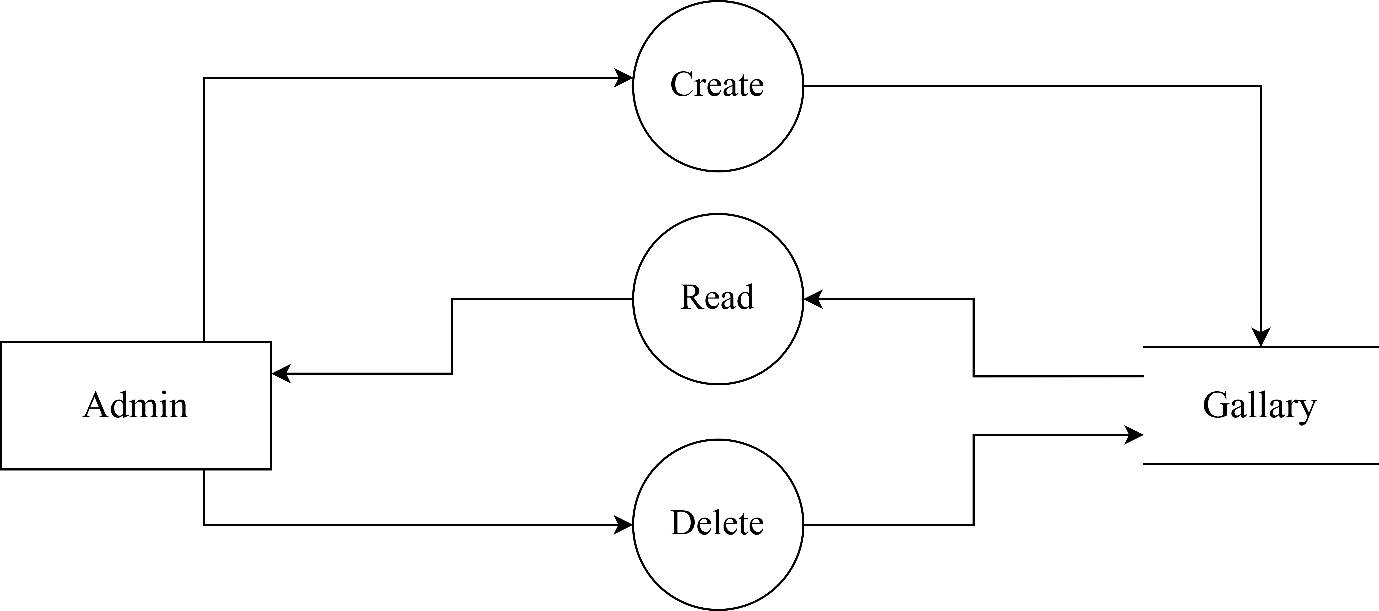
Facility level 1



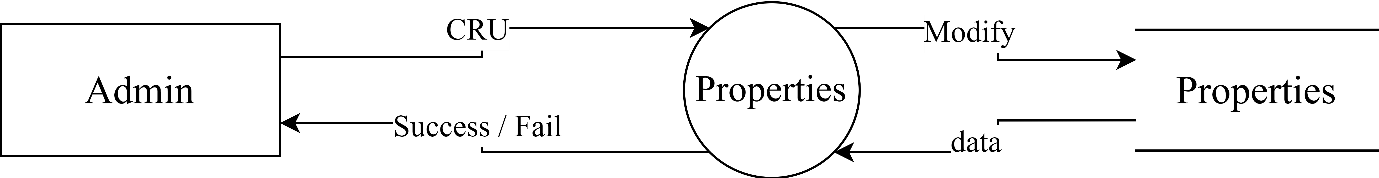
Gallery level 0



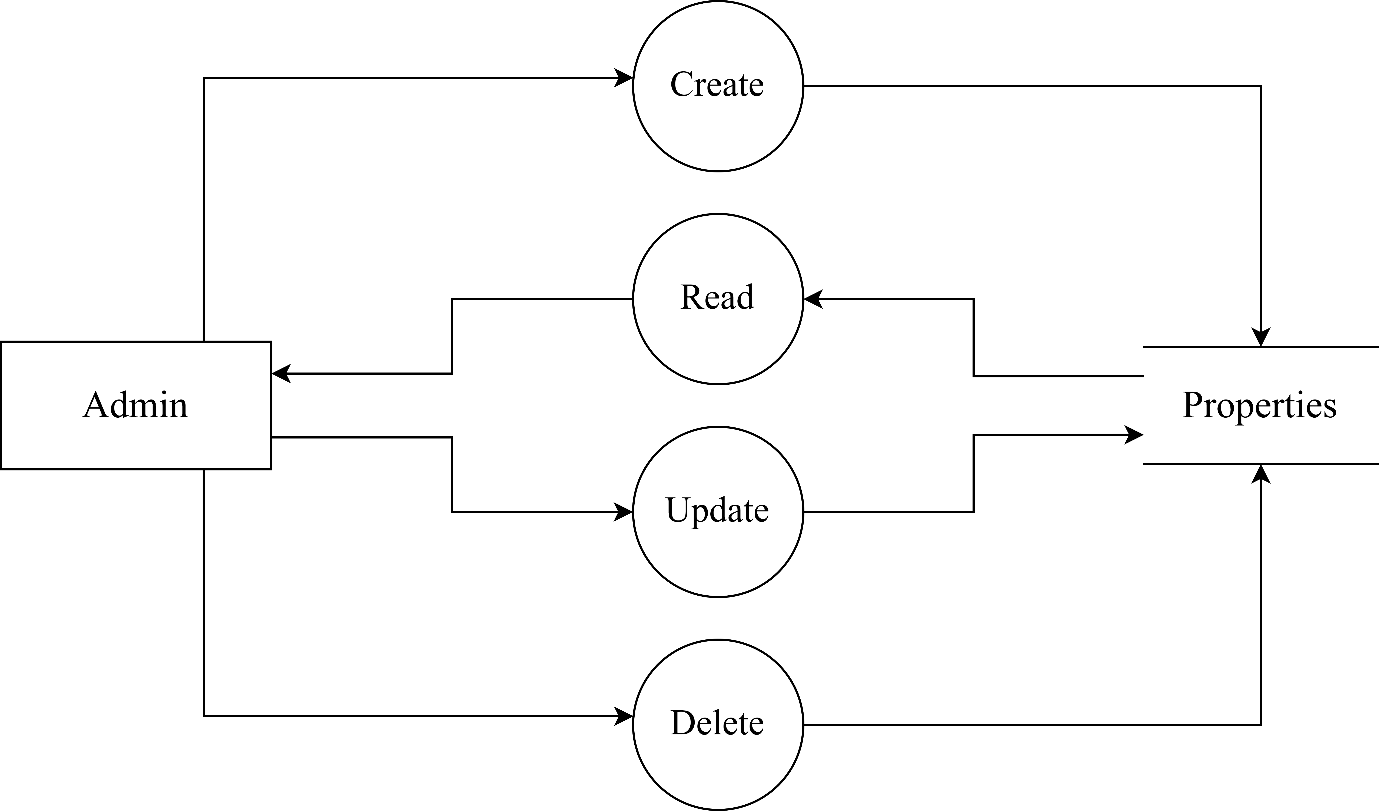
Gallery level 1



Property level 0



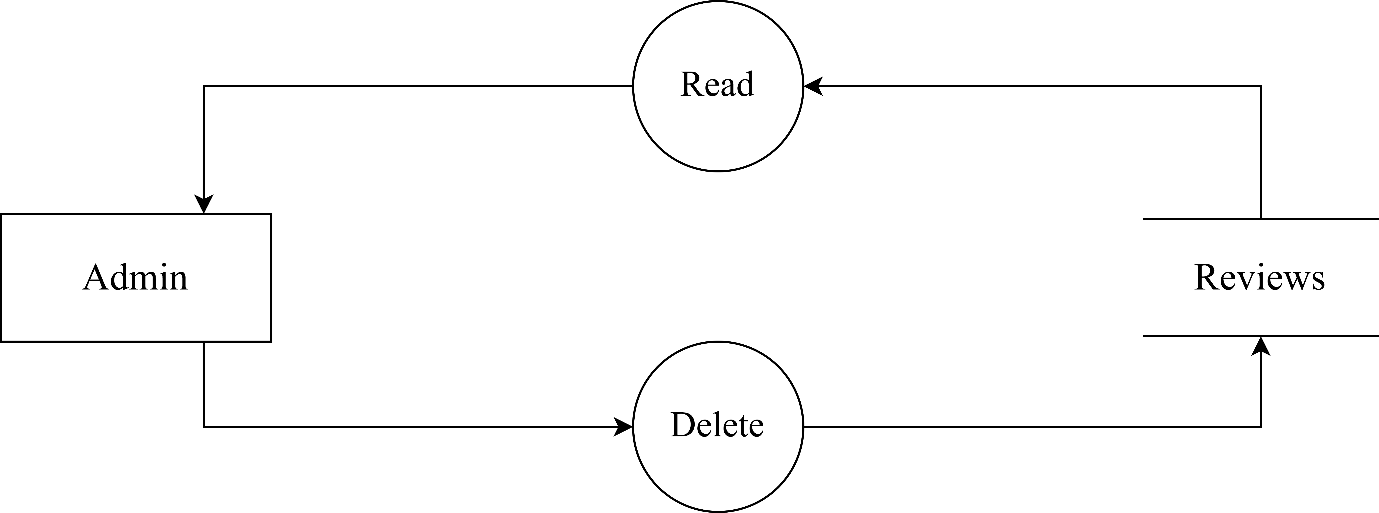
Property level 1



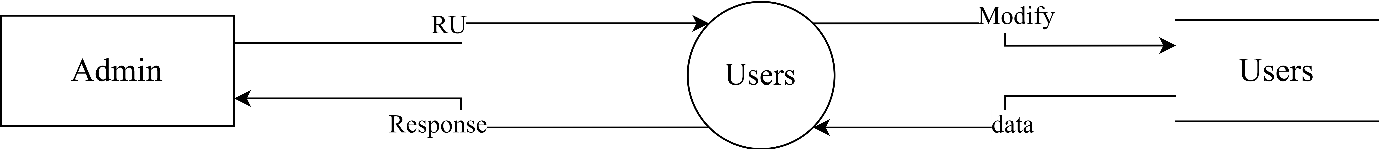
Reviews level 0



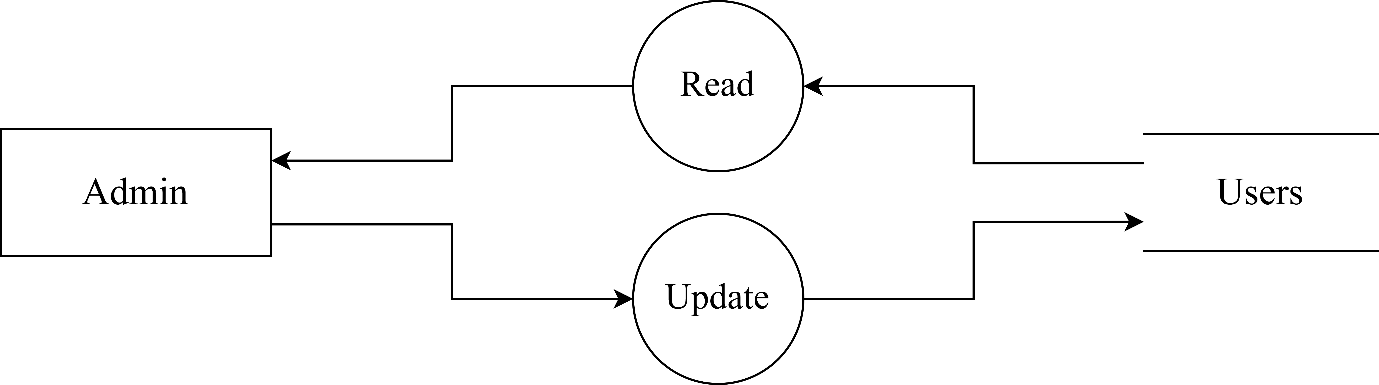
Reviews level 1



Users level 0



Users level 1

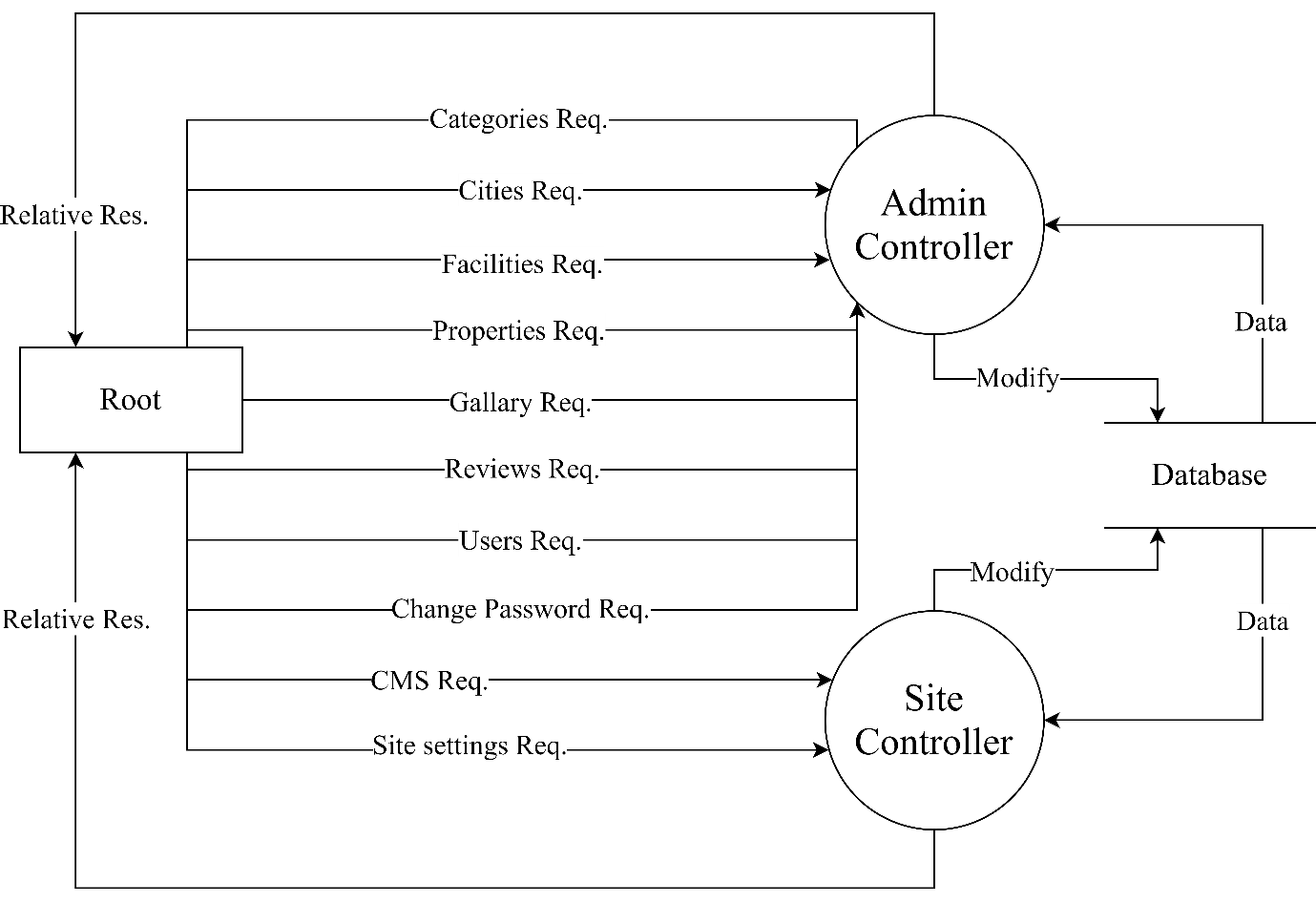


### Root DFDs

Root level 0



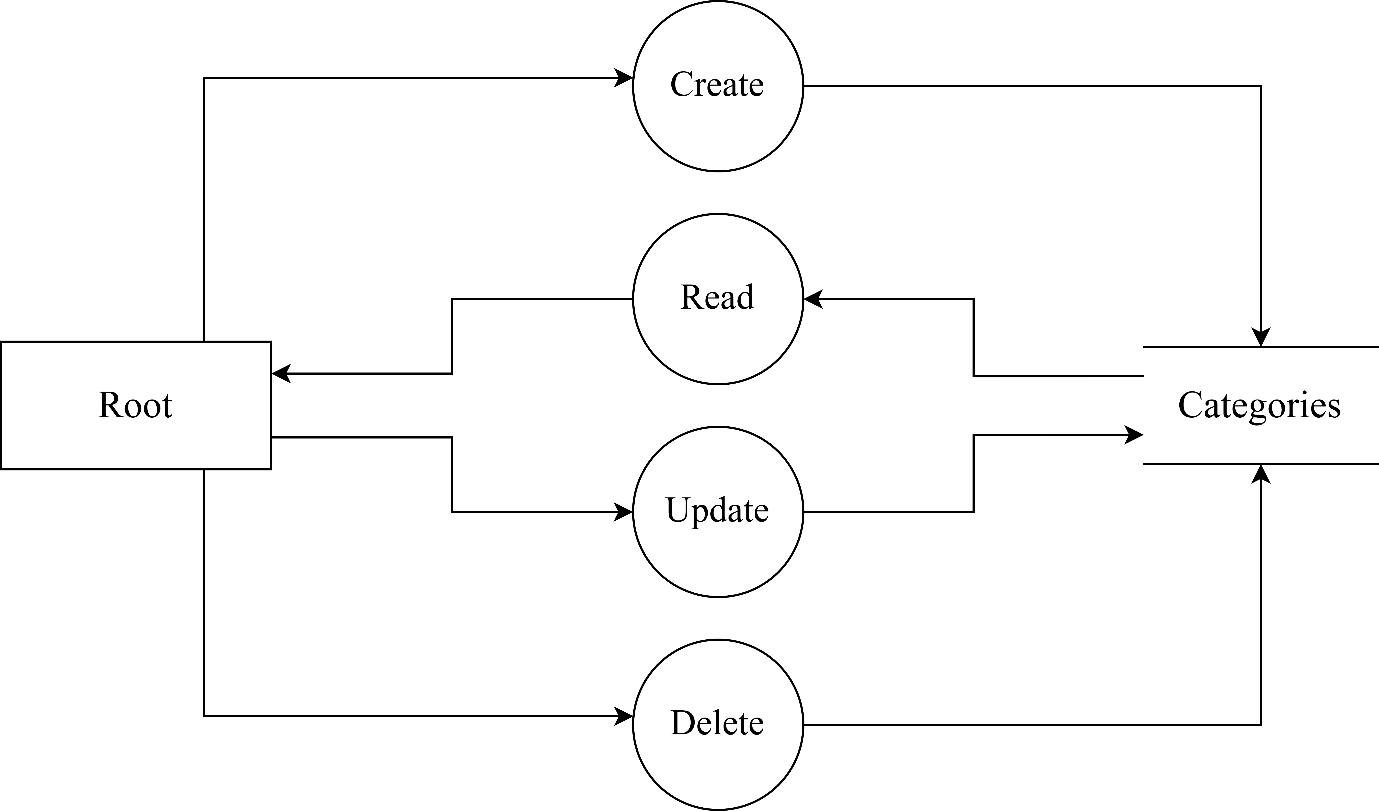
Root level 1



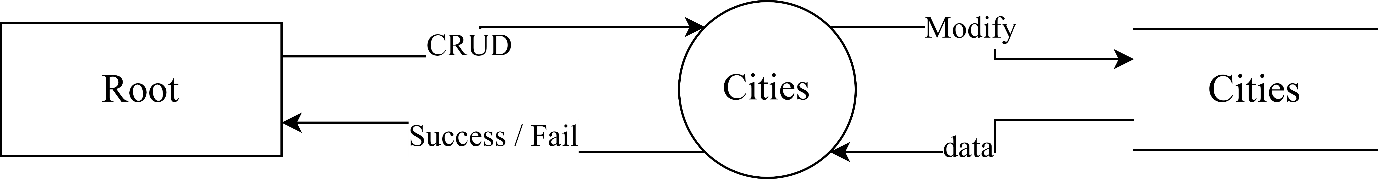
Category level 0



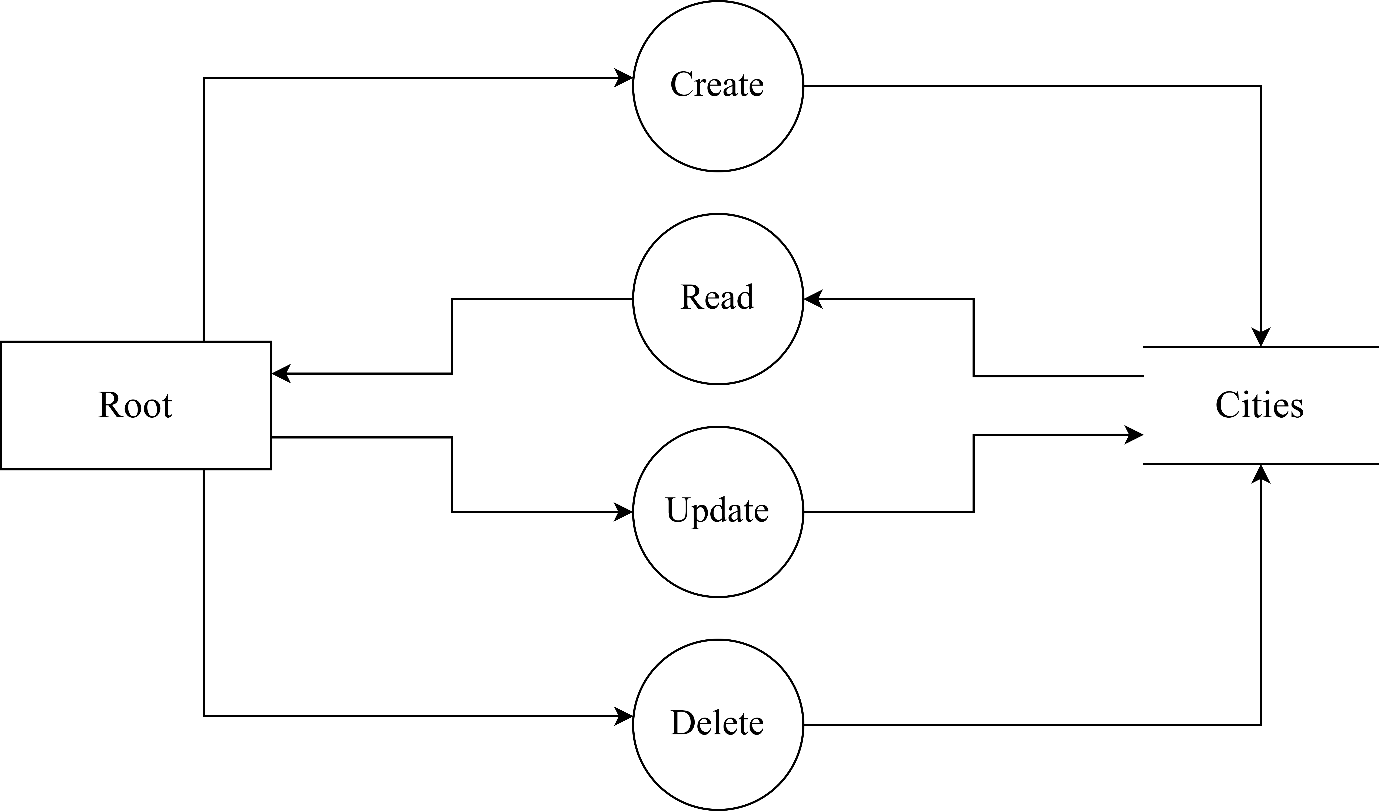
Category level 1



City level 0



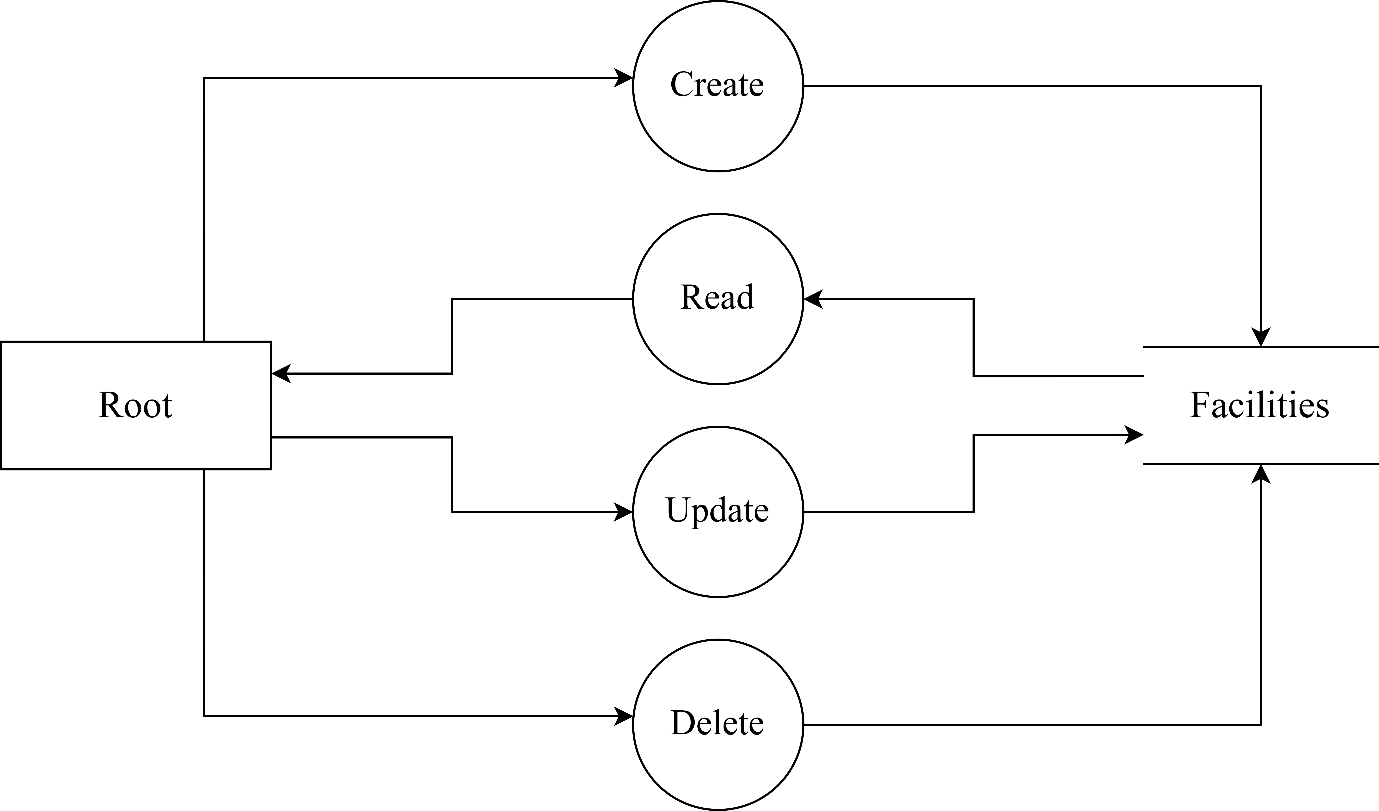
City level 1



Facility level 0



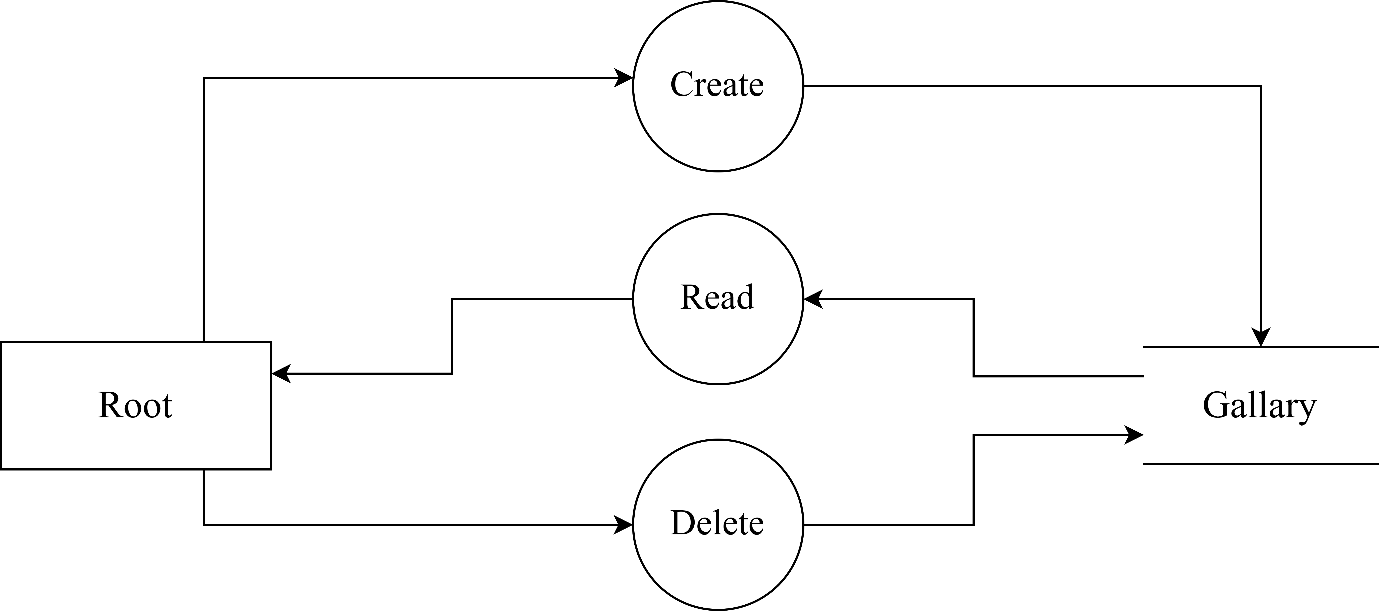
Facility level 1



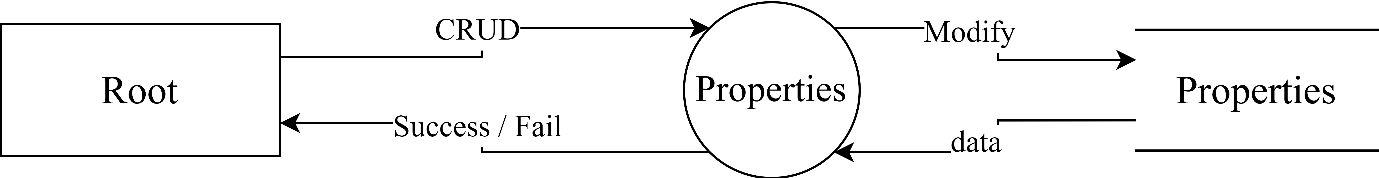
Gallery level 0



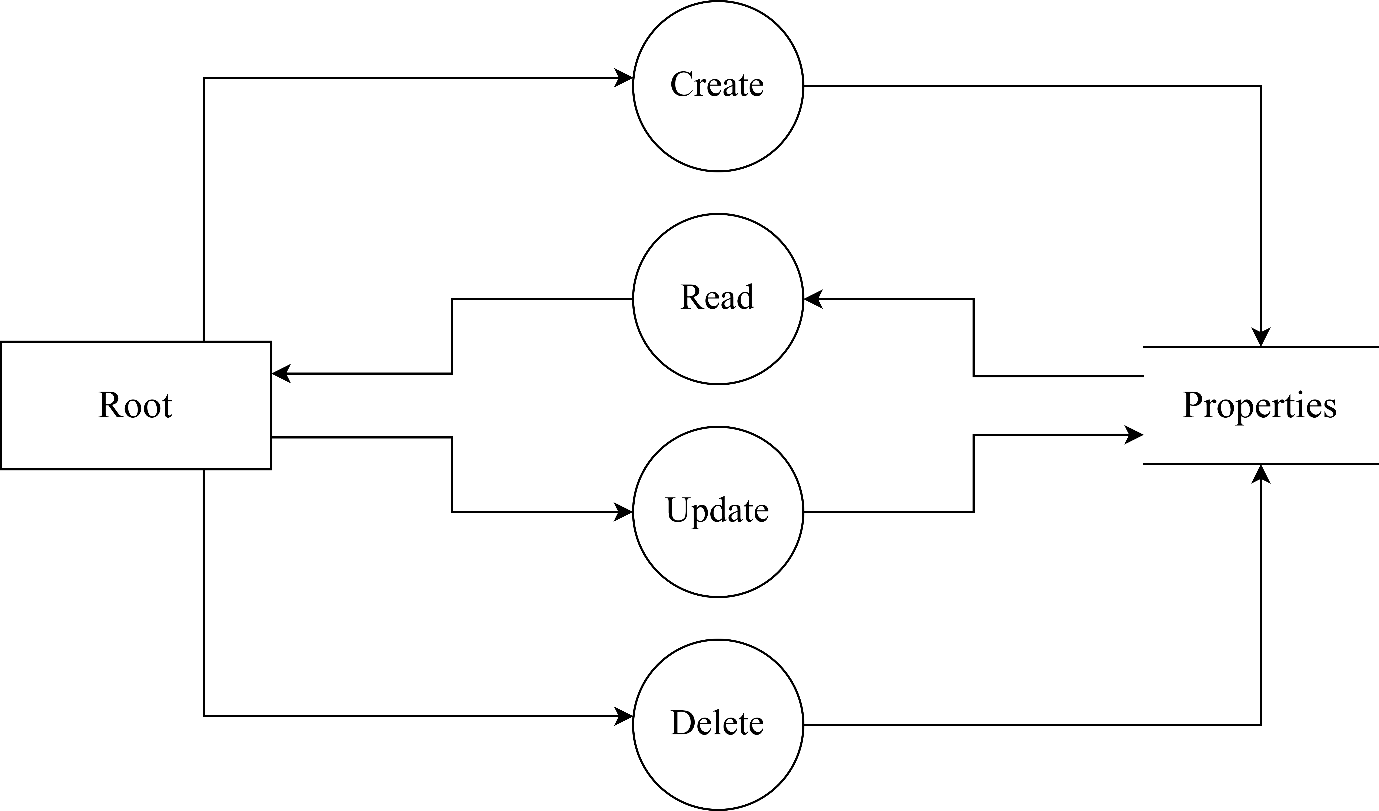
Gallery level 1



Property level 0



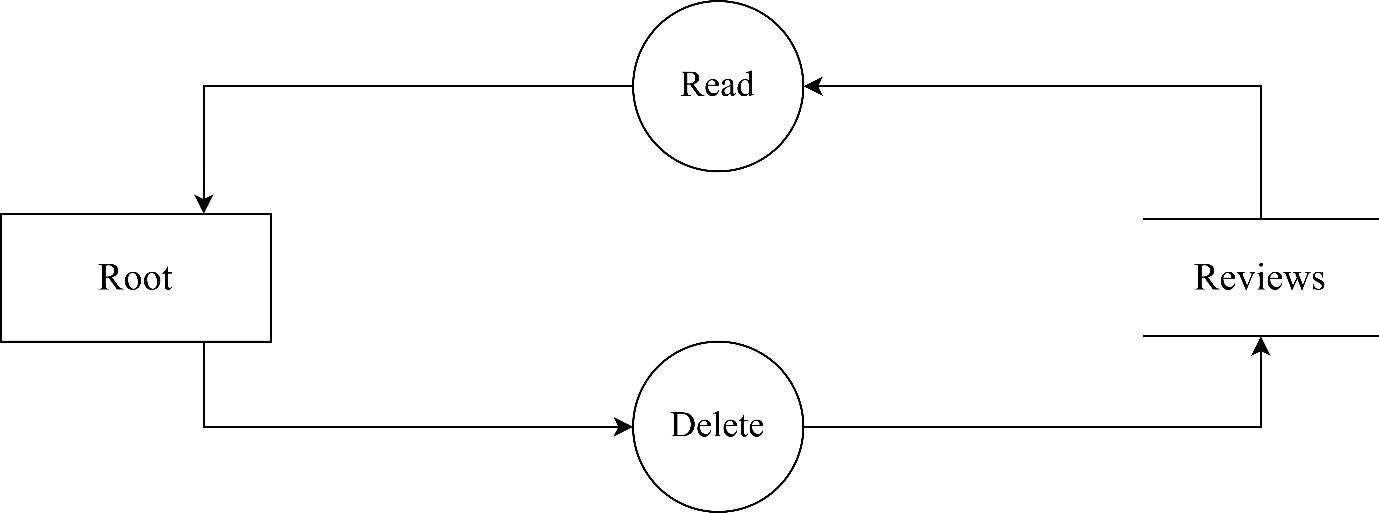
Property level 1



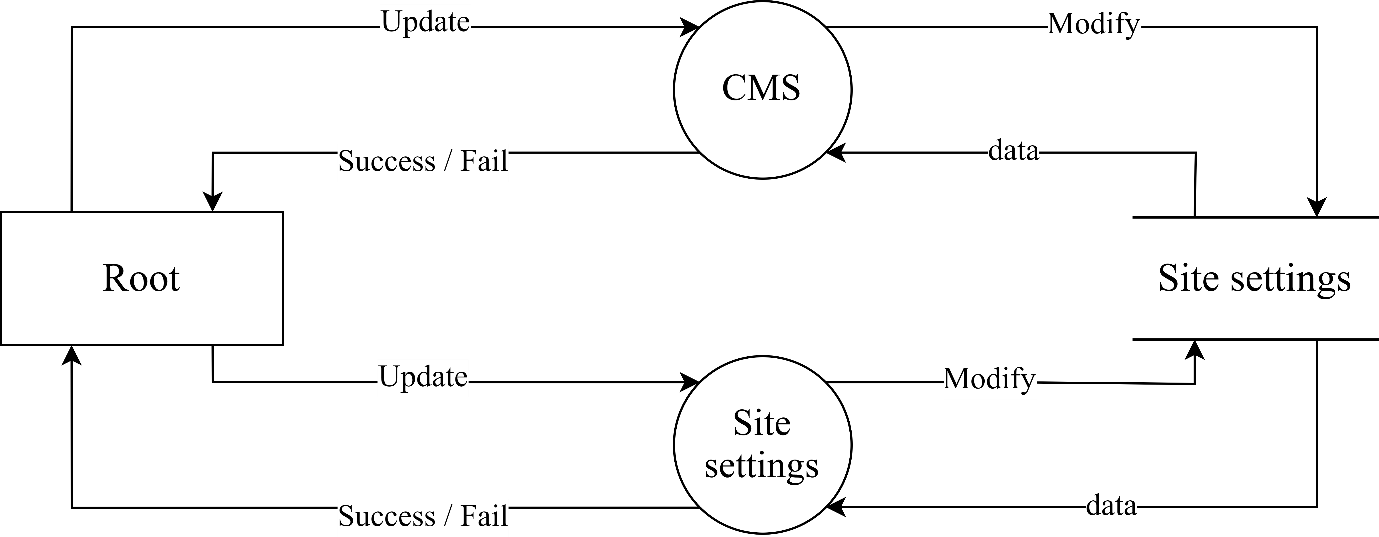
Review level 0



Reviews level 1



Site Settings level 0

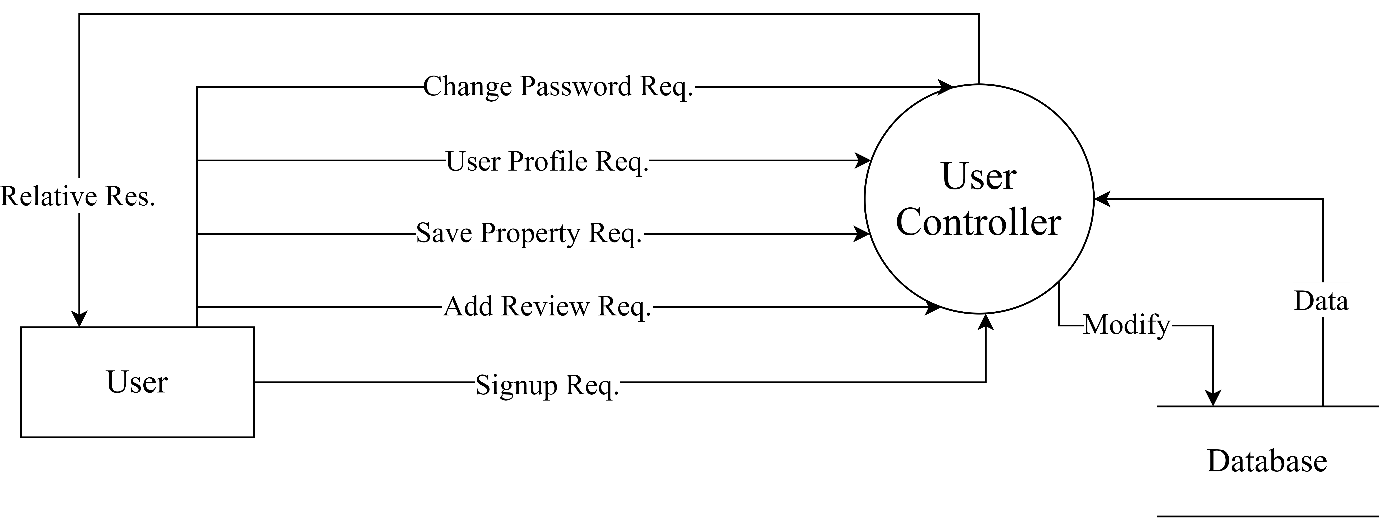


### User DFDs

User level 0



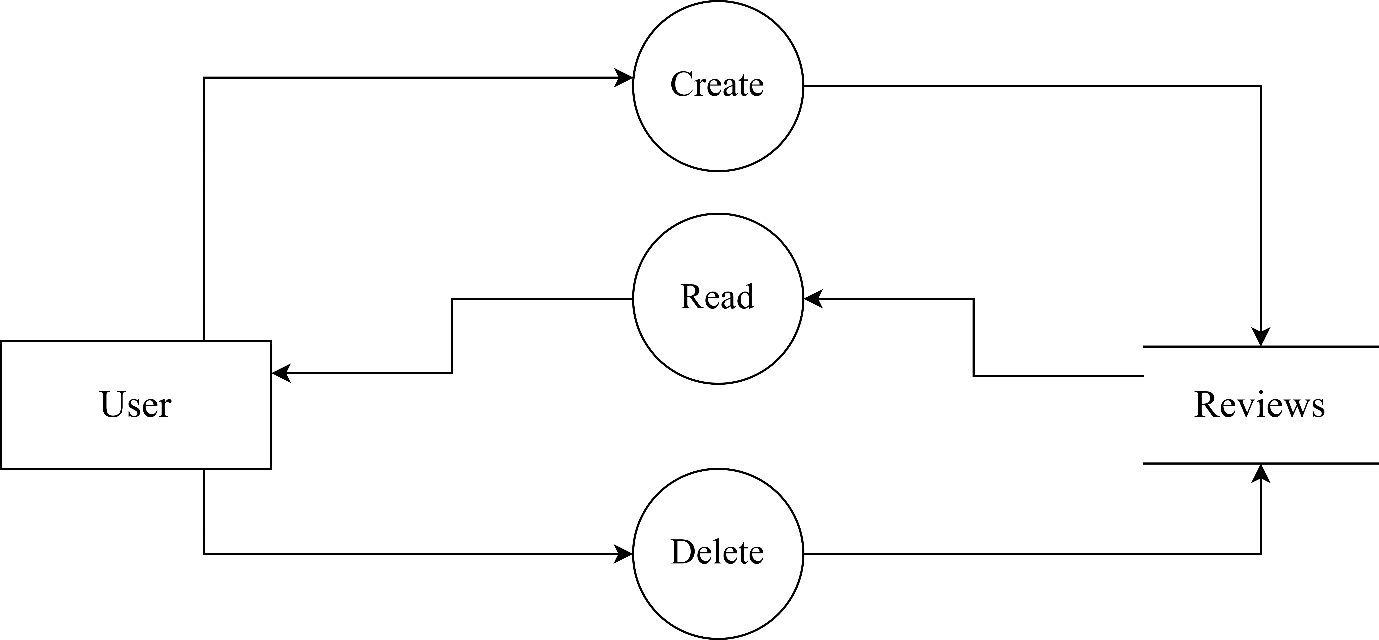
User level 1



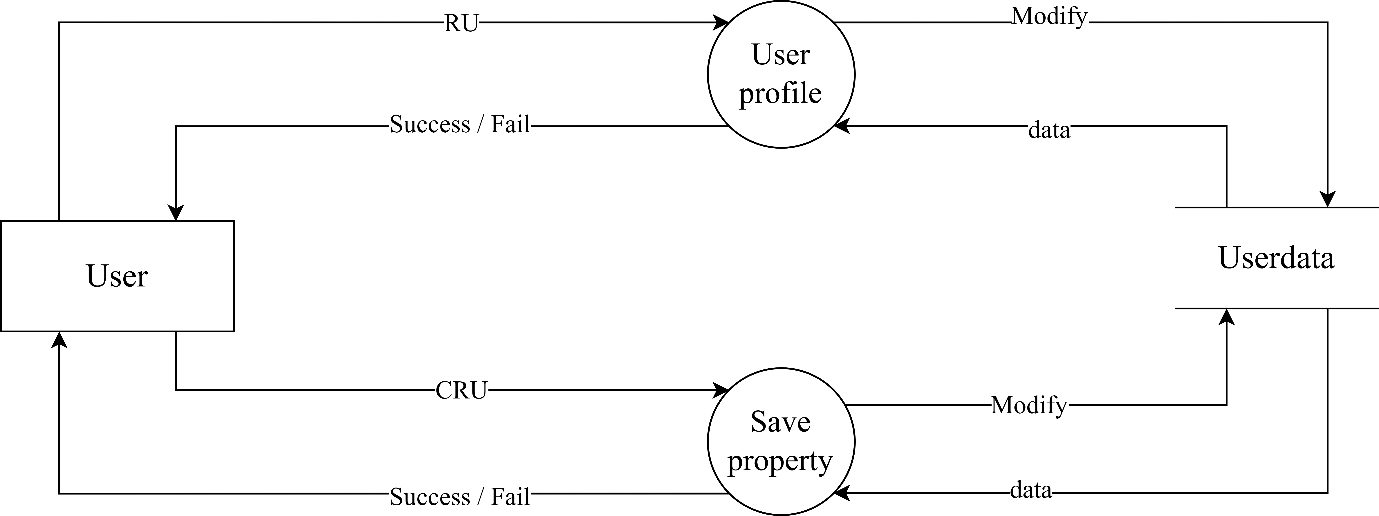
Review level 0



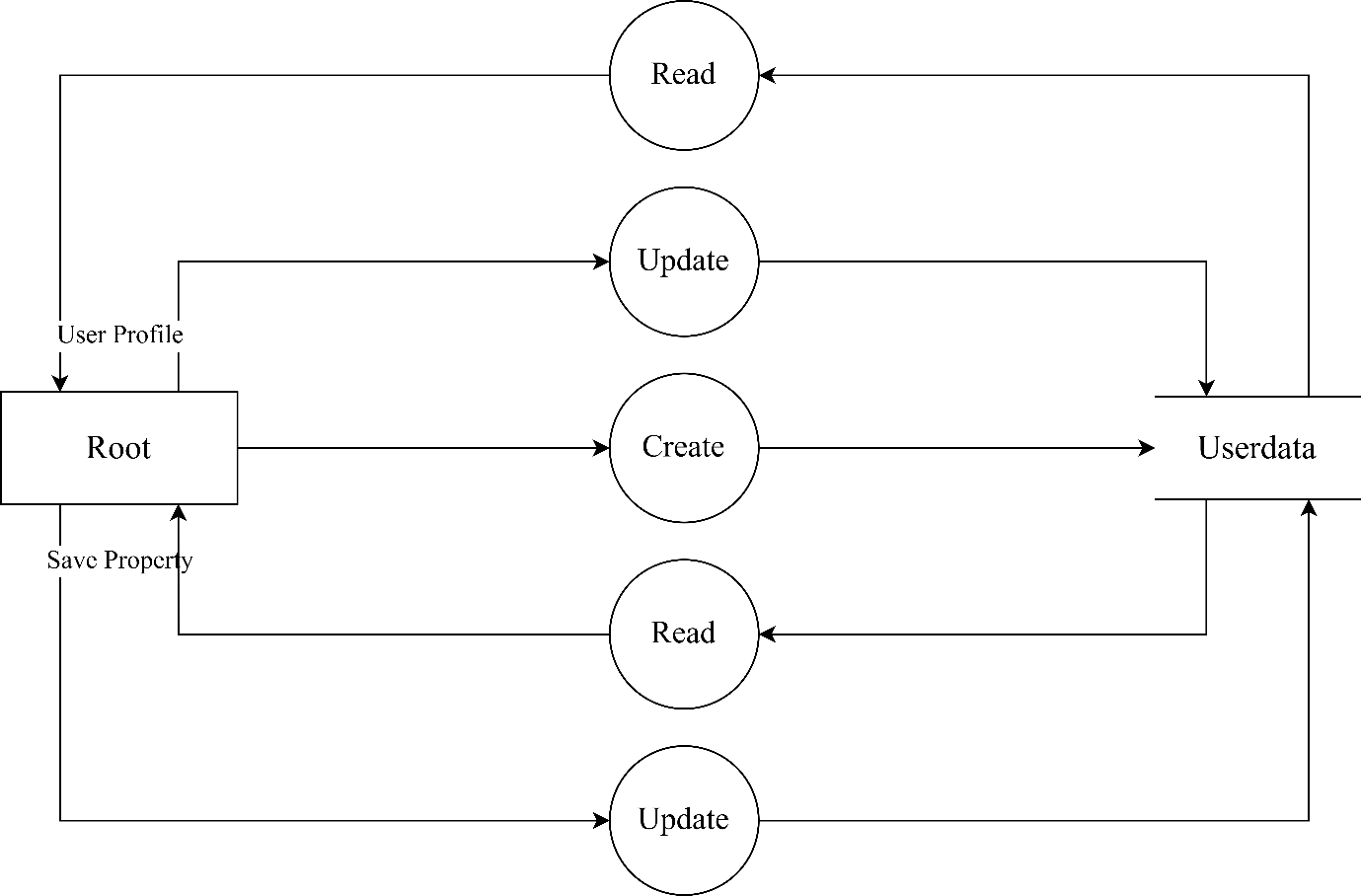
Review level 1



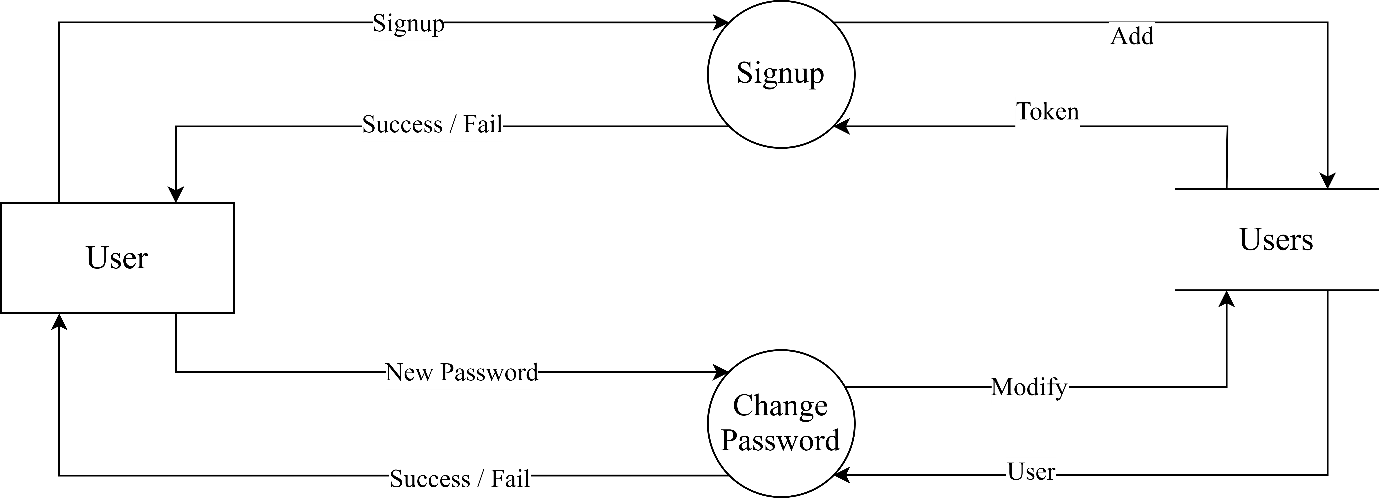
User Data level 0



User Data level 1



Users level 0



## Database Structure

Every Table will have default created\_at and updated\_at timestamps.

Categories Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Type | Null | Extra |
| PK | Id | Bigint (20) | No | AI |
| UK | Name | Varchar | No |  |
|  | Slug\_name | Varchar | No |  |
|  | Image | Varchar | No |  |

Cities Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Type | Null | Extra |
| PK | Id | Bigint (20) | No | AI |
| UK | City | Varchar | No |  |
|  | Slug\_city | Varchar | No |  |
|  | Status | Enum (‘0’, ‘1’) | No | Default (‘0’) |

CMS Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Type | Null | Extra |
| PK | Id | Bigint (20) | No | AI |
| UK | Key | Varchar | No |  |
|  | Value | Text | Yes |  |

Facilities Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Type | Null | Extra |
| PK | Id | Bigint (20) | No | AI |
| UK | Faci | Varchar | No |  |
| FK | Slug\_faci | Varchar | No |  |
|  | Fa | Varchar | Yes |  |
|  | Color | Varchar | Yes |  |

Gallaries Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Type | Null | Extra |
| PK | Id | Bigint (20) | No | AI |
| FK | Pro\_id | Bigint (20) | No |  |
|  | Gal\_image | Varchar | No |  |

Properties Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Type | Null | Extra |
| PK | Id | Bigint (20) | No | AI |
|  | Public | Tinyint (1) | No | Default (1) |
|  | Title | Varchar | No |  |
|  | Title\_slug | Varchar | No |  |
|  | Price | Decimal (10, 0) | No |  |
|  | Featured | Tinyint (1) | No |  |
|  | Purpose | Enum (‘sale’, ‘rent’, ‘pg’) | No |  |
| FK | Category | Bigint (20) | No |  |
|  | Image | Varchar | Yes |  |
|  | Fe\_image | Varchar | Yes |  |
| FK | Faci | Varchar | Yes |  |
|  | Rooms | Int | No |  |
|  | Bathrooms | Int | No |  |
| FK | City | Bigint (20) | No |  |
|  | Address | Varchar | No |  |
|  | Cont\_ph | Varchar | No |  |
|  | Cont\_em | Varchar | No |  |
|  | Area | Int | Yes |  |
|  | Description | Text | Yes |  |
|  | Video | Text | Yes |  |
|  | Floorplan | Varchar | Yes |  |
|  | Map | Text | Yes |  |

Reviews Tables

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Type | Null | Extra |
| PK | Id | Bigint (20) | No | AI |
| FK | U\_id | Bigint (20) | No |  |
| FK | Pro\_id | Bigint (20) | No |  |
|  | Review | Text | No |  |

Site Settings Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Type | Null | Extra |
| PK | Id | Bigint (20) | No | AI |
| UK | Key | Varchar | No |  |
|  | Value | Text | Yes |  |

Users Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Type | Null | Extra |
| PK | Id | Bigint (20) | No | AI |
|  | Name | Varchar | No |  |
| UK | Email | Varchar | No |  |
|  | Password | Varchar | No |  |
|  | Type | Enum (‘U’, ‘R’, ‘A’) | No |  |

User\_data Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Name | Type | Null | Extra |
| PK | Id | Bigint (20) | No | AI |
|  | Image | Varchar | Yes |  |
|  | About | Varchar | Yes |  |
|  | Saved | Text | Yes |  |

## Entity Relation Diagram

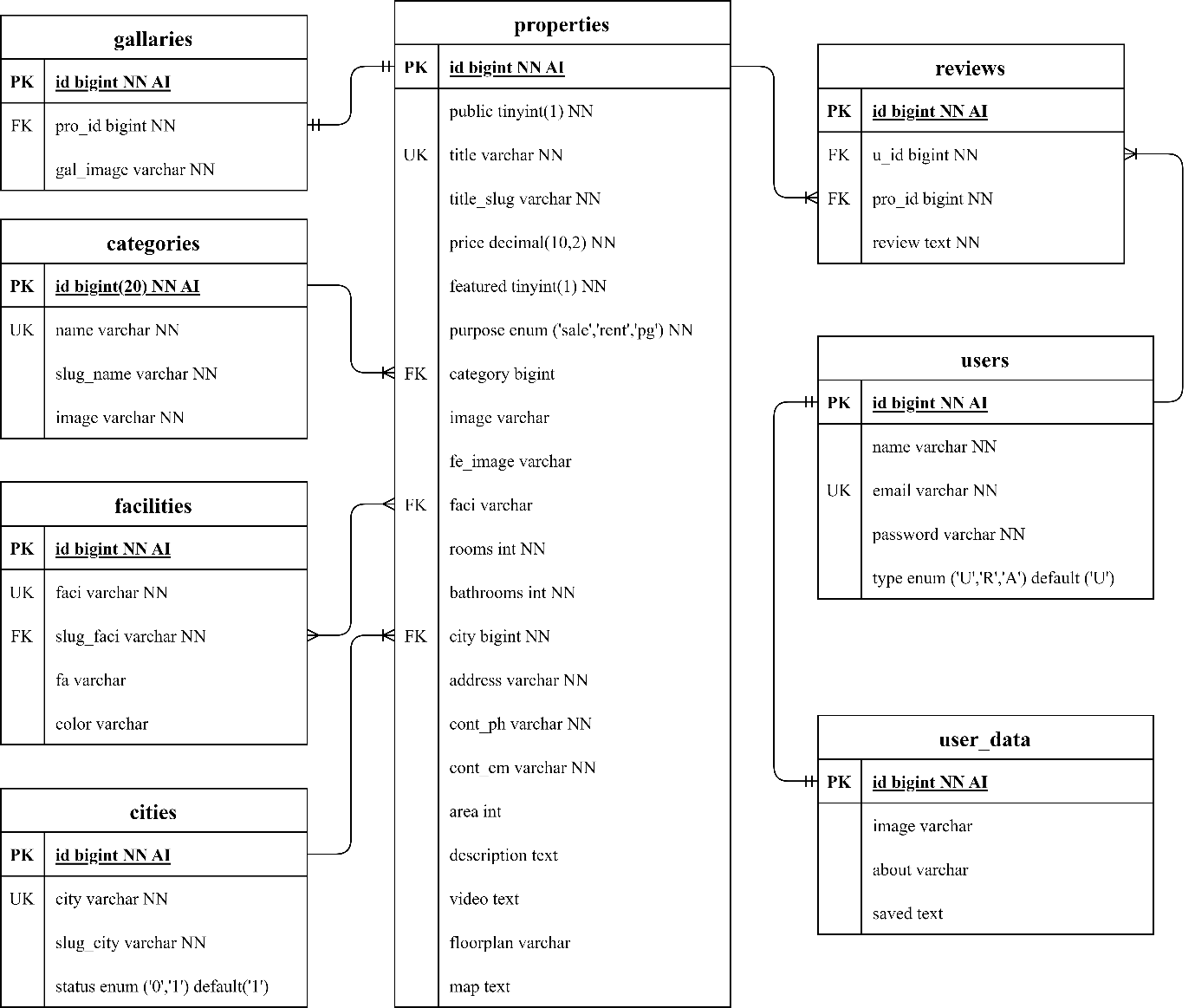
An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system.

ER diagrams are related to data structure diagrams (DSDs), which focus on the relationships of elements within entities instead of relationships between entities themselves. ER diagrams also are often used in conjunction with data flow diagrams (DFDs), which map out the flow of information for processes or systems.

Symbols and Notations Used in Crow’s Foot ERD Style (here)

|  |  |
| --- | --- |
| Notation | Symbol |
| Entity |  |
| One to One |  |
| One to Many |  |
| Many to Many |  |

ER Diagram



## Process Specification

A process specification is a method used to document, analyze and explain the decision-making logic and formulas used to create output data from process input data. Its objective is to flow down and specify regulatory/engineering requirements and procedures.

Analysis Report

# Design Report

# Testing Reports

# Limitations of the System

# Future Enhancement of the project

# References

1. Because I’m introvert I didn’t really ask someone else to help me actually. [↑](#footnote-ref-1)
2. Because I only learn things when I’m under pressure. [↑](#footnote-ref-2)
3. Fun fact: every semester there was one new faculty replacing other. [↑](#footnote-ref-3)