System Design Document

Project: Secure Web App

By: The Cheesecake Hub

# Architecture

## Web Framework

**Django** is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support.

Django helps you write software that is:

**Complete:** Django follows the "Batteries included" philosophy and provides almost everything developers might want to do "out of the box". Because everything you need is part of the one "product", it all works seamlessly together, follows consistent design principles, and has extensive and up-to-date documentation.

**Versatile:** Django can be (and has been) used to build almost any type of website — from content management systems and wikis, through to social networks and news sites. It can work with any client-side framework and can deliver content in almost any format (including HTML, RSS feeds, JSON, XML, etc.). Internally, while it provides choices for almost any functionality you might want (e.g. several popular databases, templating engines, etc.), it can also be extended to use other components if needed.

**Secure**: Django helps developers avoid many common security mistakes by providing a framework that has been engineered to "do the right things" to protect the website automatically. For example, Django provides a secure way to manage user accounts and passwords, avoiding common mistakes like putting session information in cookies where it is vulnerable (instead cookies just contain a key, and the actual data is stored in the database) or directly storing passwords rather than a password hash.

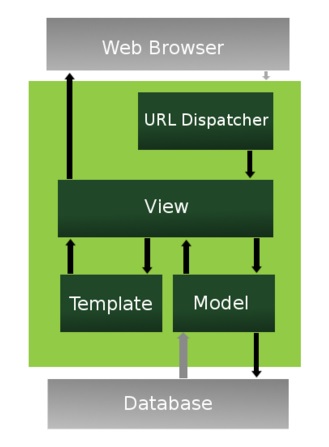
Django enables protection against many vulnerabilities by default, including SQL injection, cross-site scripting, cross-site request forgery and clickjacking (see Website security for more details of such attacks).

**Scalable**: Django uses a component-based “shared-nothing” architecture (each part of the architecture is independent of the others and can hence be replaced or changed if needed). Having a clear separation between the different parts means that it can scale for increased traffic by adding hardware at any level: caching servers, database servers, or application servers. Some of the busiest sites have successfully scaled Django to meet their demands (e.g. Instagram and Disqus, to name just two).

**Maintainable**: Django code is written using design principles and patterns that encourage the creation of maintainable and reusable code. In particular, it makes use of the Don't Repeat Yourself (DRY) principle so there is no unnecessary duplication, reducing the amount of code. Django also promotes the grouping of related functionality into reusable "applications" and, at a lower level, groups related code into modules (along the lines of the Model View Controller (MVC) pattern).

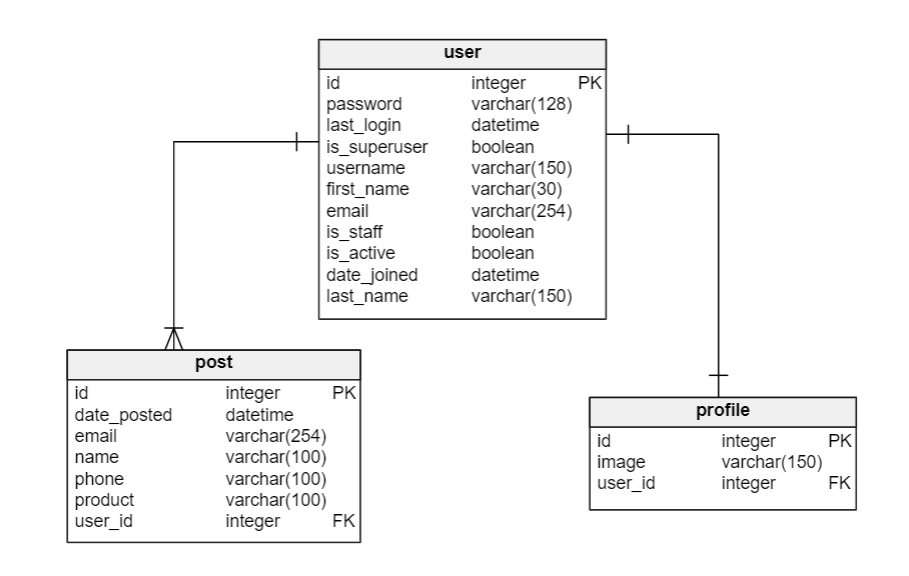
**Portable**: Django is written in Python, which runs on many platforms. That means that you are not tied to any particular server platform, and can run your applications on many flavours of Linux, Windows, and Mac OS X. Furthermore, Django is well-supported by many web hosting providers, who often provide specific infrastructure and documentation for hosting Django sites.

(Source: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction)

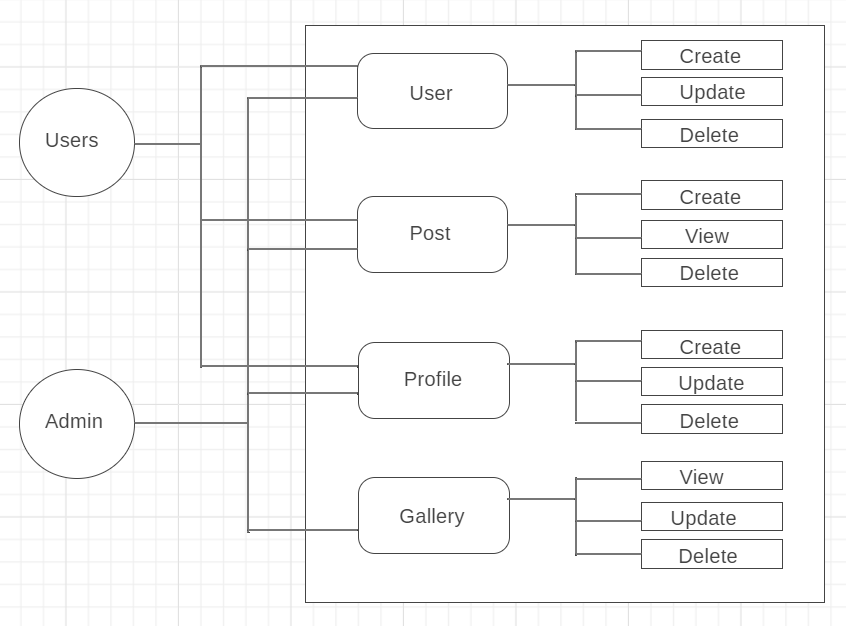


# Data Access Design

## Model Design (Data Model)

**

## Functional Decomposition



## Database Investigation

Django in its 'out-of-the-box' state is set up to communicate with **SQLite** -- a lightweight relational database included with the Python distribution. So by default, Django automatically creates a SQLite database for your project.

In addition to SQLite, Django officially supports (i.e. included in Django itself) three other popular relational databases that include: PostgreSQL, MySQL and Oracle. And unofficially (i.e. with third party packages) Django supports connectivity to other relational databases that include: SAP (Sybase) SQL Anywhere, IBM DB2 and Firebird, as well as the ADO (ActiveX Data Objects) and ODBC (Open Database Connectivity) interfaces, the last two of which are standard for connecting to Microsoft SQL Server and the latter is supported by most relational database brands.

# Security Design

## Framework Security

**User Authentication**

Django comes with a user authentication system. It handles user accounts, groups, permissions and cookie-based user sessions. The Django authentication system handles both authentication and authorization. Briefly, authentication verifies a user is who they claim to be, and authorization determines what an authenticated user is allowed to do. Here the term authentication is used to refer to both tasks.

The auth system consists of:

* Users
* Permissions: Binary (yes/no) flags designating whether a user may perform a certain task.
* Groups: A generic way of applying labels and permissions to more than one user.
* A configurable password hashing system
* Forms and view tools for logging in users, or restricting content
* A pluggable backend system

## Security Mechanisms

Security procedures included:

* Protect the system from common security threats and attacks.
* Protect user information and product orders.

User passwords will be encrypted automatically by Django.

## User Roles and Permissions

* Administrators: Full access (Create, Update, Delete)
* Users: Create only (product orders)

# User Interface Design

## View Design (UI List)

### User Story 1

As a café owner I would like to be able to securely create an account so I may view cheesecake recipes and place product orders.

Views:

* Register

### User Story 2

As a café owner I would like to privately send cheesecake suppliers correspondence and product orders so that I can securely and discreetly purchase product for my cafe.

Views:

* PostCreateView (product order)

## UI Design

**Hex colour palette:**

--blue: #007bff

--indigo: #6610f2

--purple: #6f42c1

--pink: #e83e8c

--red: #dc3545

--orange: #fd7e14

--yellow: #ffc107

--green: #28a745

--teal: #1abc9c

--cyan: #17a2b8

--white: #fff

--gray: #6c757d

--gray-dark: #343a40

--primary: #1abc9c

--secondary: #2c3e50

--success: #28a745

--info: #17a2b8

--warning: #ffc107

--danger: #dc3545

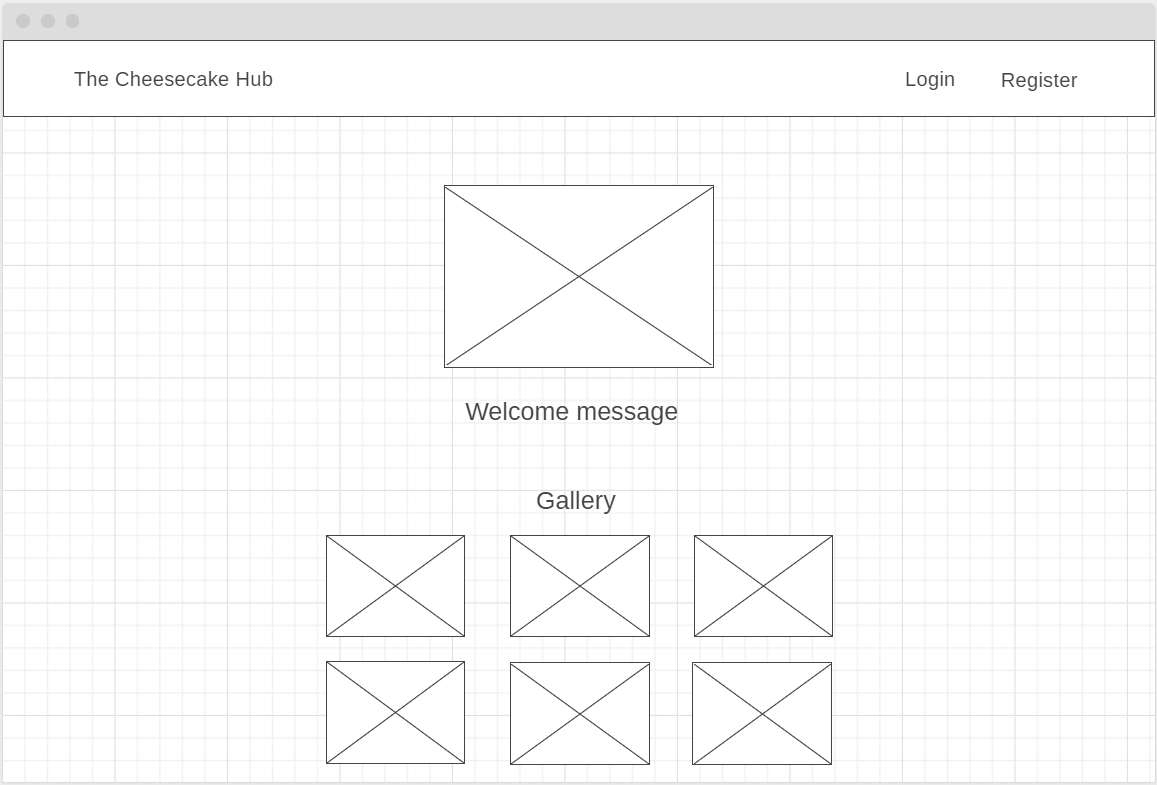
--light: #f8f9fa

--dark: #343a40

**Fonts:**

* Late
* Segoe UI
* Helvetica Neue
* Noto Sans
* Liberation Mono
* Courier New

**Wireframe (home page):**



Home page continued…

