Design Document

CS 4503 - Team Deadpool

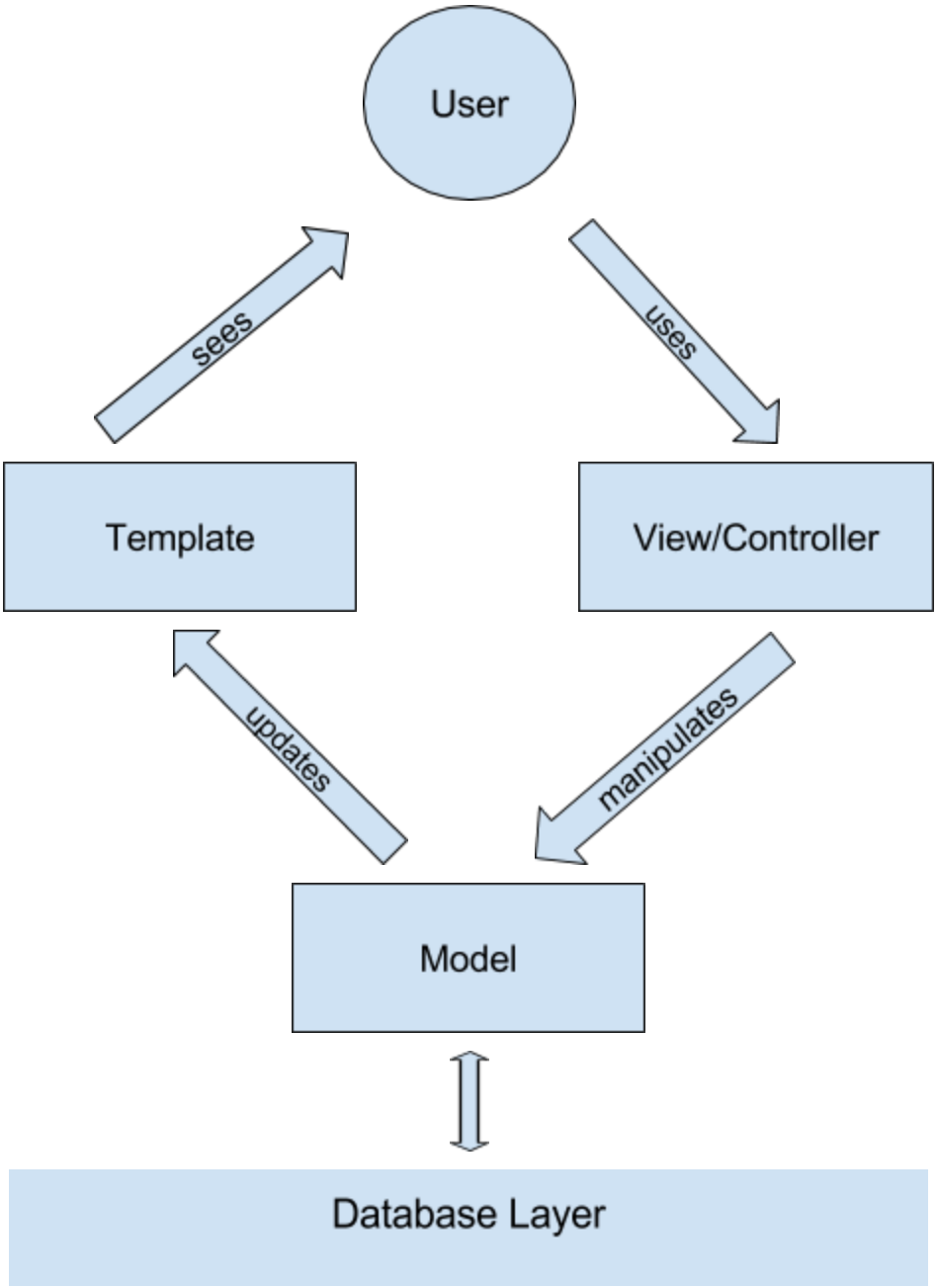
Zachary, Mike, Sam, Raymond, Kyle, Phil

1. **Introduction**

   This document will lay out the high level design of the Heartland Gaming Expo web application. This document includes architecture diagrams as well as entity - relation diagrams depicting the database schema. This document is intended for software engineers, system architects, and all others involved in implementing and maintaining the Heartland Gaming Expo system.

2. **Architecture**

   2.1 Introduction



The architectural model of the Heartland Gaming Expo system is the MVC model. The lowest layer in the model is the Database Layer which will consist of a database that stores all persistent app information. The top three layers will be provided by the Django framework. The top three layers provide mapping from database tables to Python objects (Django models), manipulation of those Python objects (Django views/controllers), and displaying those Python objects along with other UI components to the user (Django templates).

2.2 Modules

2.2.1 Database Layer

The database layer is responsible for holding all the persistent data for the system and defining the relationships between this data. The system will be using a database that can be easily integrated with Django. The data relationships are described in more detail below in section 3.1.

2.2.2 Model Layer - Django Models

This layer is responsible for mediating all access to the database. This layer maps the database tables and fields into Python objects/Django models which will be manipulated through the Logic and GUI layers. The primary functions of this layer involves querying the database and mapping queries to Python objects. This layer will also have functions to create new database entries or update existing entries based on changes from the GUI and Logic components.

2.2.3 Controller/View Layer - Django Views

This layer is responsible for performing necessary tasks with data from the database. The views is used by the user to access the Model layer and thus access the database. The controller mediates the flow of data between the database and the user. To clarify, Django Views correlate to the Controller layer in the traditional MVC architecture.

2.2.4 GUI Layer - Django Templates

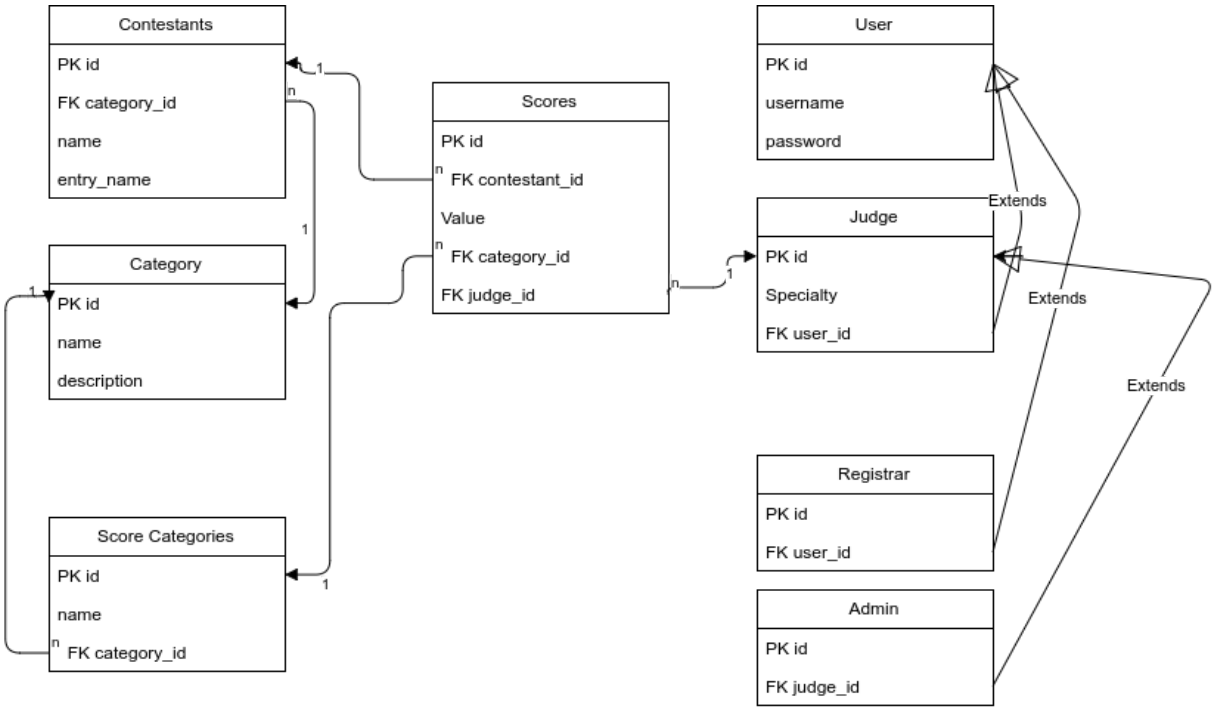
This layer is responsible for providing specific HTML webpages to users to create a GUI interface. It consists of several different templates that can be displayed to a user depending upon the user's role and actions. This layer does not handle business logic, but performs client-side checks to provide useful feedback to users, and passes user interaction to the controllers/views for processing. It is updated by the Model layer to display the correct template to the user. To clarify Django Templates correspond to the View layer in the traditional MVC architecture

3. Class Diagrams

3.1 Data Table Classes

The system will be using a database to hold all data for the judging system. This includes information on contestants, users, scores, and categories, as well as relationships between those entities. The figure below shows each table in the database with its respective columns and as well as the relationships between the different tables.

3.1.1 Schema



3.1.2 Schema Information

The data in the database will be organized into the following tables and columns

Contestants: Holds information for the contestants in the Heartland Gaming Expo. Includes an ID and the category that the contestant is entered in. This table also includes a name for the contestant.

Category: This table contains information about the categories that contestants participate in, including the name of the category and a description (e.g. Middle School, High School, Code Jam, etc.). This table also contains the scoring categories/criteria for each category.

Score Categories: This table contains information about the specific score categories including a name and description of the category (e.g. Graphics, Creativity, etc.)

Scores: This table contains the actual value of a specific score category for a specific team. This table also contains a foreign key which is tied to the judge that input the score.

User: This table contains the usernames and passwords for the users of the system, namely Judges, Administrators, and Registrars.

Judge: This table extends the User table and includes information about Judges such as specialty, and QR code used for login

Registrar: This table extends the User table and contains a foreign key which is tied to the user table.

Admin: This table extends the Judge table and contains a foreign key which is tied to the judge table.

3.2 Class Information

Classes will be implemented using Django models. The models will replicate the database schema in section 3.1.1, with each table mapping to a Django model. The data in these models will be presented to the user through the View and Template layers. The Logic and GUI layers consist of Django controller/view classes and templates which determine the way data is manipulated and presented to the user.

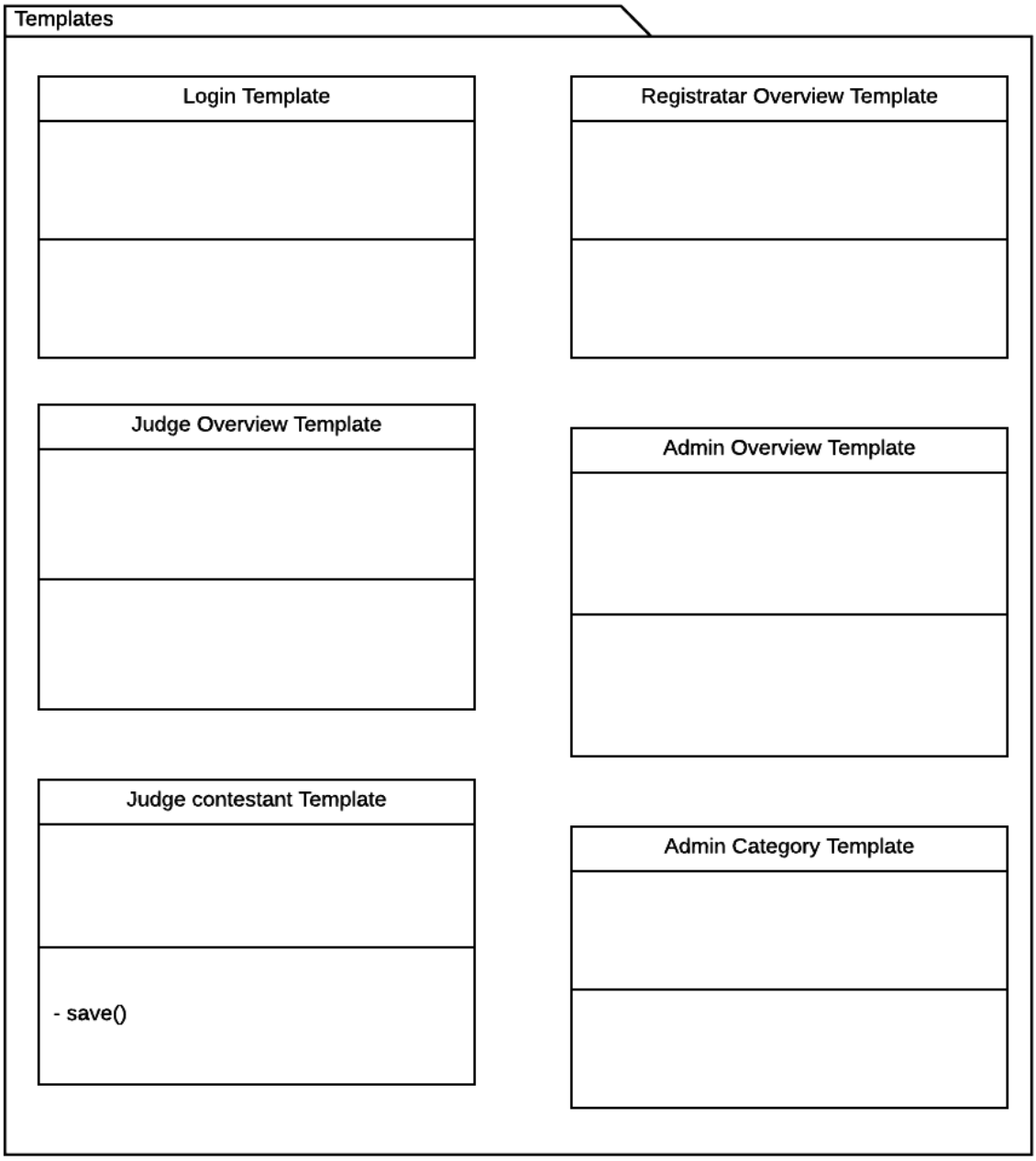
3.3 Template Layer

The template layer of the system contains html templates to be dynamically constructed by data from the view layer and displayed to the user. The template later provides the user interface of the system in a web GUI fashion.

The first template the user will see is the Login Template, where the user will input login information in the form of user/password or QR scan to proceed to his/her specific overview template. If the user is classified as a judge, he/she will be taken to the Judge Overview template, which displays a sortable list of contestants to choose. Upon selection of a team, the judge will be taken to the Judge Contestant template which displays the contestant's relevant scoring criteria and allows the judge to submit or edit his/her scores for the contestant and then redirect the user to the Judge Overview template.

If the user is a registrar, he/she will be taken to the Registrar Overview template. The Registrar Overview template displays a list of teams already registered with an option to

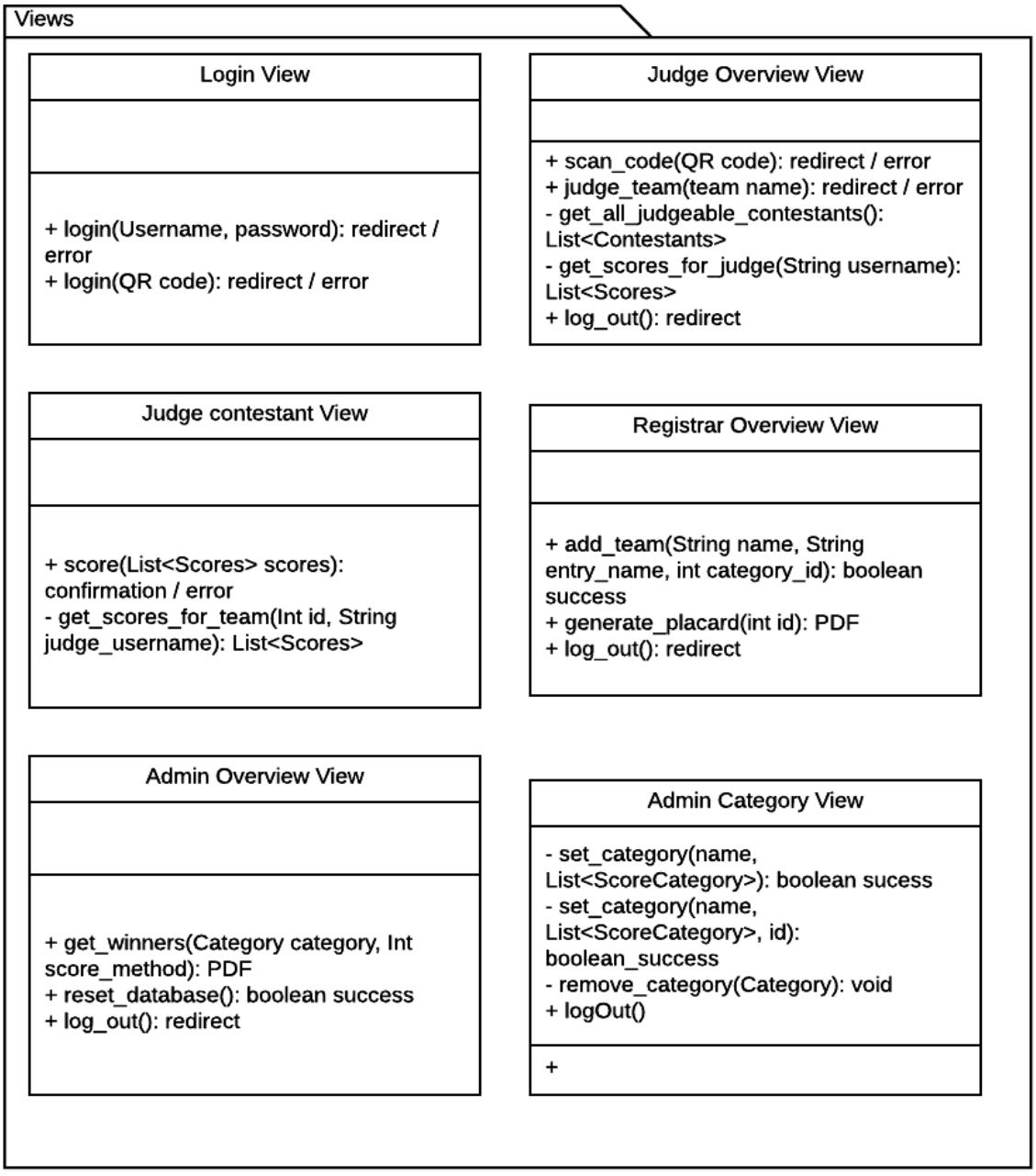
generate and print a team's QR code. The Registrar Overview will also allow the user to add a team to the contestant table and refresh the list.
If the user is an administrator, the user will be directed to the Admin Overview which contains the various functions and operations necessary to administration. Functions such as getting scores, resetting the database, performing registrar and judging operations, and redirection to the Admin Category template. The Admin Category template provides a list of categories and score-categories to manage by edit/removal/deletion.



3.4 View Layer

The view layer consists of various classes which are used by the user to manipulate the model and by extension the database. There will be one controller class for every template. The views determine what data should be retrieved from the database and displayed to the user through the templates. There will be view classes for each user role

which control the various actions performed by each user. The view classes associated with the Judge will control entering scores for teams. The Registrar view will control registering new teams. The Administrator view will be responsible for adding/deleting categories, getting the overall score results, and flushing the data from the database.



3.5 Model Layer

The model layer of the system contains Python classes which corresponds to each table in the schema presented in Section 3.1.1. The Judge and Registrar classes extend the User class. The Admin class extends the Judge class. The class hierarchy for Users is used to clearly define the access level of each type of user. The model layer will also contain two interfaces that are implemented by the User subclasses. The Judge class implements the Judge interface, the Registrar class implements the Registrar interface, and the Administrator class implements both the Judge and Registrar interfaces.

The rest of the classes in the Model layer are used to manage contestants and their scores. The Category class contains the categories that contestants participate in, i.e. High School, Middle School, Code Jam, etc. The Score Categories class represents the scoring criteria within each category, i.e. Graphics, Playability, Awesomeness, etc. The Scores class is used to record the actual score given for each score category. The Contestant class manages the data for each contestant in the Expo.