Software Architecture Document

For the Better Betting Pool App

By Team C:
Will Blankenship, Ryan Manus, and David Rosenberg
Team contact e-mail: dcrsnbrg@memphis.edu

Submitted
November 17, 2017

Introduction

This document is intended to provide the reader with some high-level architecture models of the Better Betting Pool, a Django-based betting pool app being developed as a student software engineering project exercising agile software development methods. Three different models will be examined: client-server system architecture, the Django infrastructure, and betting pool classes.

Client-Server System Architecture

The Better Betting Pool is a web-based application, whose functionality is delivered by a Django server, which generally follows a client-server architectural pattern across a network. In the case of the Better Betting Pool, the stand-alone Django server provides web, application, and database services, which clients can interface by web, across a network infrastructure. This architectural model is shown in figure 1:

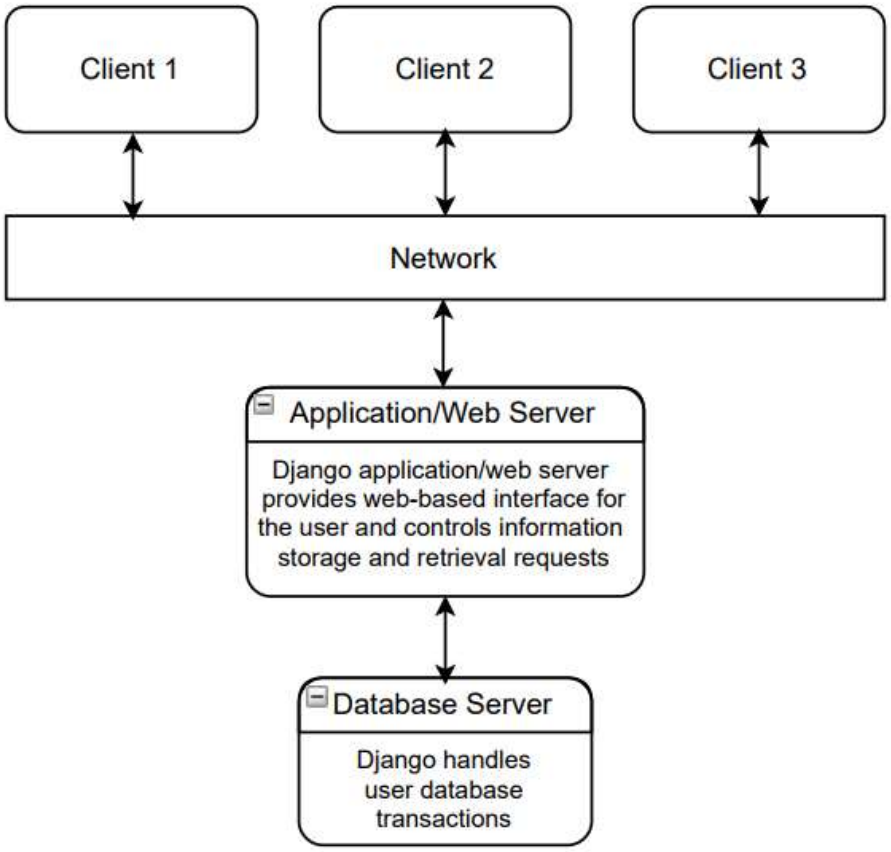
Client 1
Client 2
Client 3
Network
Application/Web Server
Django application/web server provides web-based interface for the user and controls information storage and retrieval requests
Database Server
Django handles user database transactions

Figure 1: Client-server architecture of the Better Betting Pool app

Django Infrastructure

Django is a Python web framework, which supports the MVC (model-view-controller) pattern, a web-based system where the user interface is implemented using a web browser. Django follows MVC with a slight variation: the controller component is handled by Django and the view is implemented by a template, an HTML (web) file with embedded DTL (Django Template Language). As described on tutorialspoint.com, figure 2 illustrates Django's MVC-MVT components and their interactions:

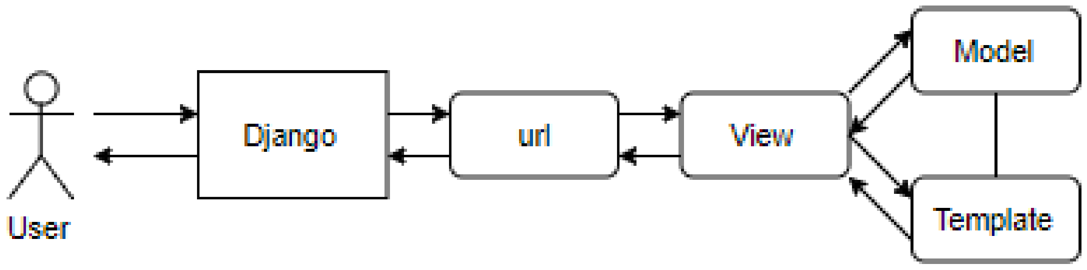

Figure 2: Django MVC-MVT pattern component interaction

Better Betting Pool Classes

After gathering some basic requirements and developing some user stories, the initial design began to take form. Each class is related either by one-to-one, foreign key, or by interaction during runtime. The class diagram shown in figure 3 details some attributes of each object as well as relationships between the objects. Cardinality (one-to-one or many-to-one—foreign key) is indicated by number, and dotted lines represent relationships that occur during runtime but not within the database.

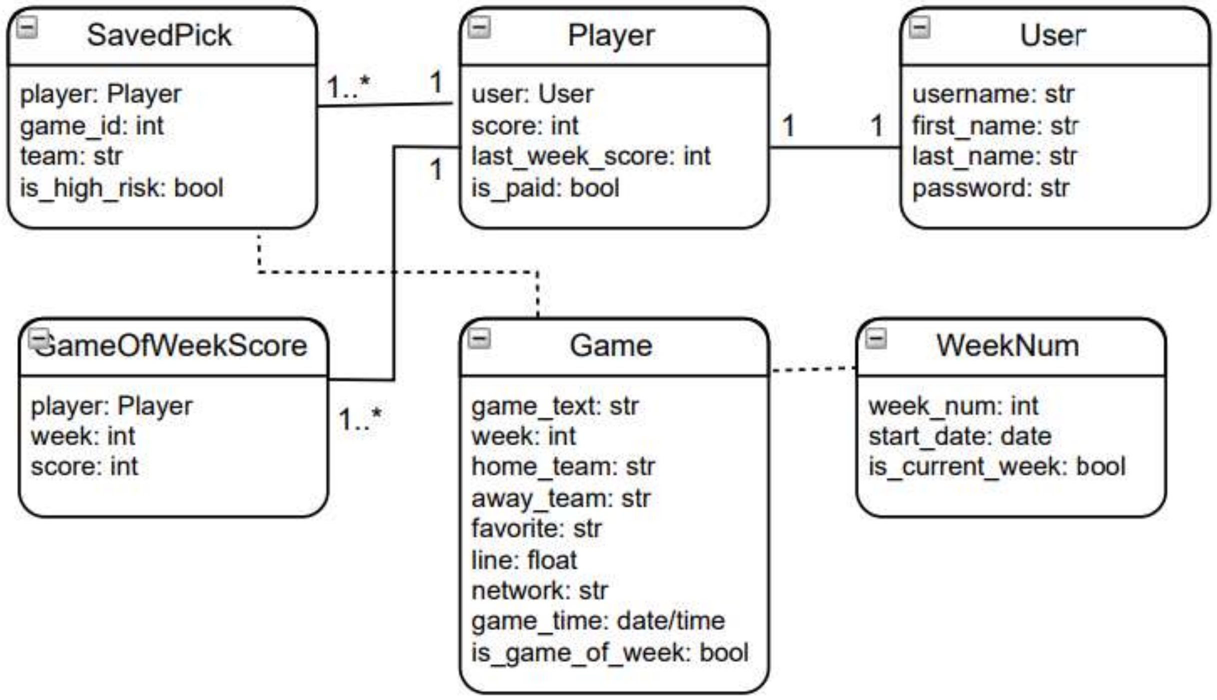

Figure 3: Better Betting Pool class diagram