ChopChop MIDI Key Interpreter Suite

SE-480: Software Architecture

Table of contents

Project Overview

Section 1: ChopChop Interpreter

Section 2: MapBuilder

Section 3: Game

Section 4: Controller

Section 5: Server

Appendices

Overview

ChopChop is a suite of applications for Windows, OSX, and Linux operating systems that interprets keystrokes from a piano-like MIDI device and converts them into keystrokes which are sent to the system. It consists of an interpreter, an application for constructing layouts to be used by the interpreter, and a game that allows a user to learn how to use the system. In addition, a server facilitates the sharing of layouts that users have constructed and interfaces with social media for sharing scores and achievements.

The Interpreter uses chording so that combinations of multiple keys on the MIDI keyboard produce a combination of keystrokes that occur on the system on which ChopChop is running. If only selecting from the twelve notes in a single octave, with each hand pressing up to five notes, there are nearly 2.5 million possible key combinations, and for each possible key combination there are millions of potential sequences of keys, leading to an essentially infinite number of combinations for layouts.

In order to help create coherent and efficient layouts from these virtually limitless possibilities a MapBuilder application is also provided. The MapBuilder performs a statistical analysis of provided samples to determine which combinations of letters occur most frequently and uses a scheme to associate them with keystrokes in such a way as to enable users to type much faster than is possible with traditional keyboards. On a QWERTY keyboard, speeds of 60 to 90 words per minute (WPM) are possible. With some chorded keyboards, speeds of 180 wpm have been achieved. With Stenographers keyboards, speeds greater than 200 wpm are common. Depending on the schemes employed, these kind of results could be expected from ChopChop as well. Additionally, the MapBuilder application validates that a generated layout includes all primitive key combinations that would be available on a standard keyboard, as well as checking for potentially malicious key combinations.

In order to help users make sense of a layout, a Game application is provided as well. The game interfaces with the application to access the frequency with which keystrokes are used. Lower levels of the game test basic keystrokes, and as the player advances, more complex patterns emerge. As the player advances in the game the words per minute they can type increase. In the game, the user plays as an organism, with other organisms that float through the field of vision. Each of these organisms has a sequence of keystrokes that are indicated as text associated with each instance, and are drawn from the layout. By successfully executing the correct sequence for an organism smaller than the player character, the player character consumes the associated character and grows. If instead the player executes the sequence associated with an organism larger than the player's character, a bite is taken from the player character. As the player progresses, their organism grows, from an amoeba to a multicellular organism, octopi of various sizes, and eventually a Kraken. Besides providing training on how to use ChopChop, the Game also provides a mechanism for user-driven optimization. The competition to achieve higher levels should lead to the development of layouts that enable faster typing.

Requirements

Interpreter

The Interpreter shall quickly process incoming MIDI signals from the selected device and produce the appropriate keystrokes.

The Interpreter shall process all triggered MIDI events from the selected device.

All possible keystrokes should be attainable in any layout used for the interpreter.

The Interpreter should have an optional guide mode which displays hints for operation.

The Interpreter should have an optional audio mode which sends incoming audio signals to a MIDI output device.

MapBuilder

Map validation should be performed to verify that all primitive keystrokes are obtainable within a layout.

Map validation should be performed to verify that no malicious sequences exist within a layout.

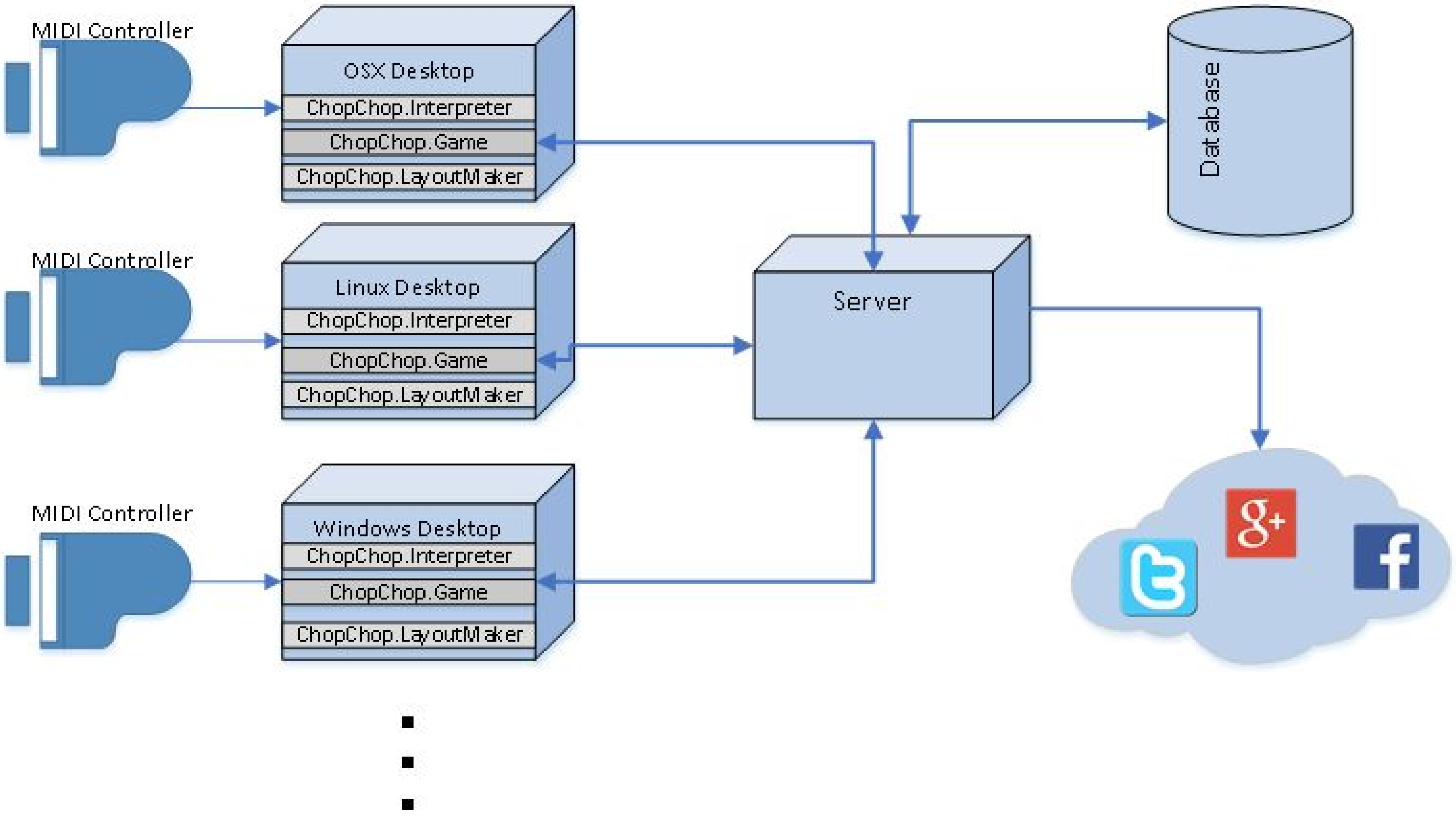
The MapBuilder should be able to produce layouts optimized for a variety of purposes.

The MapBuilder should be able to produce layouts in a variety of languages

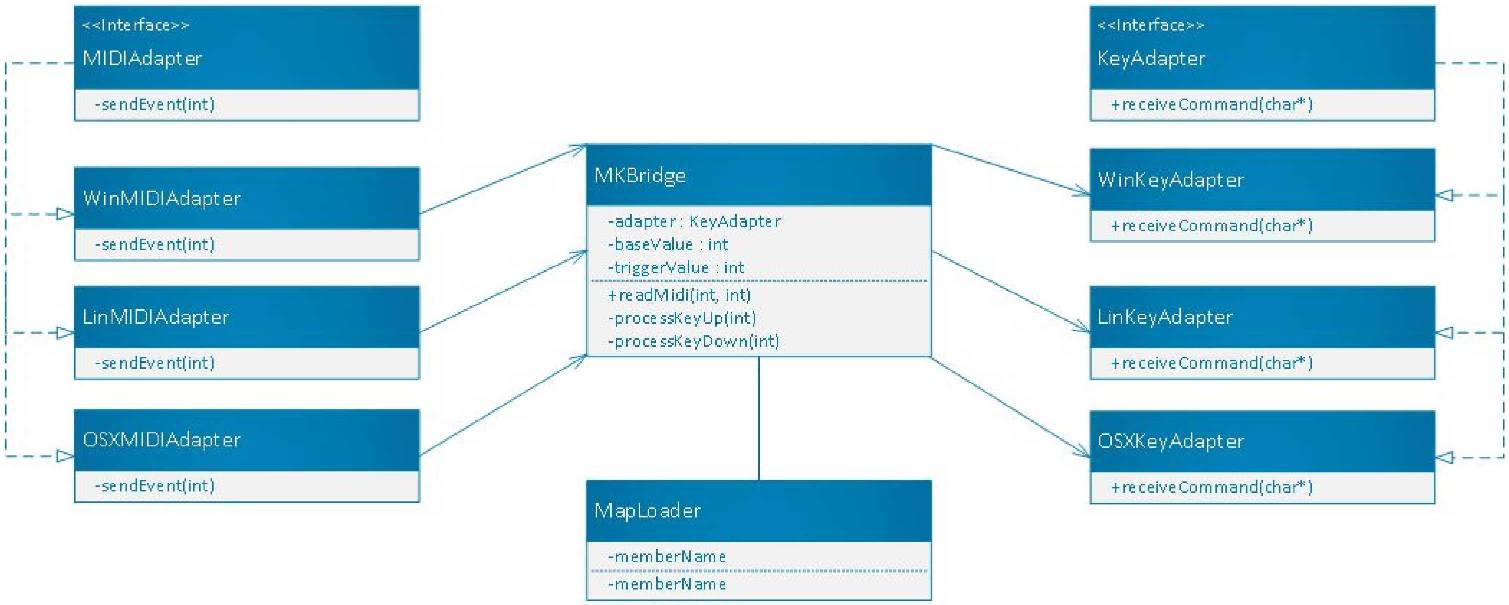
Game

The game should provide incremental training on use of the selected layout.

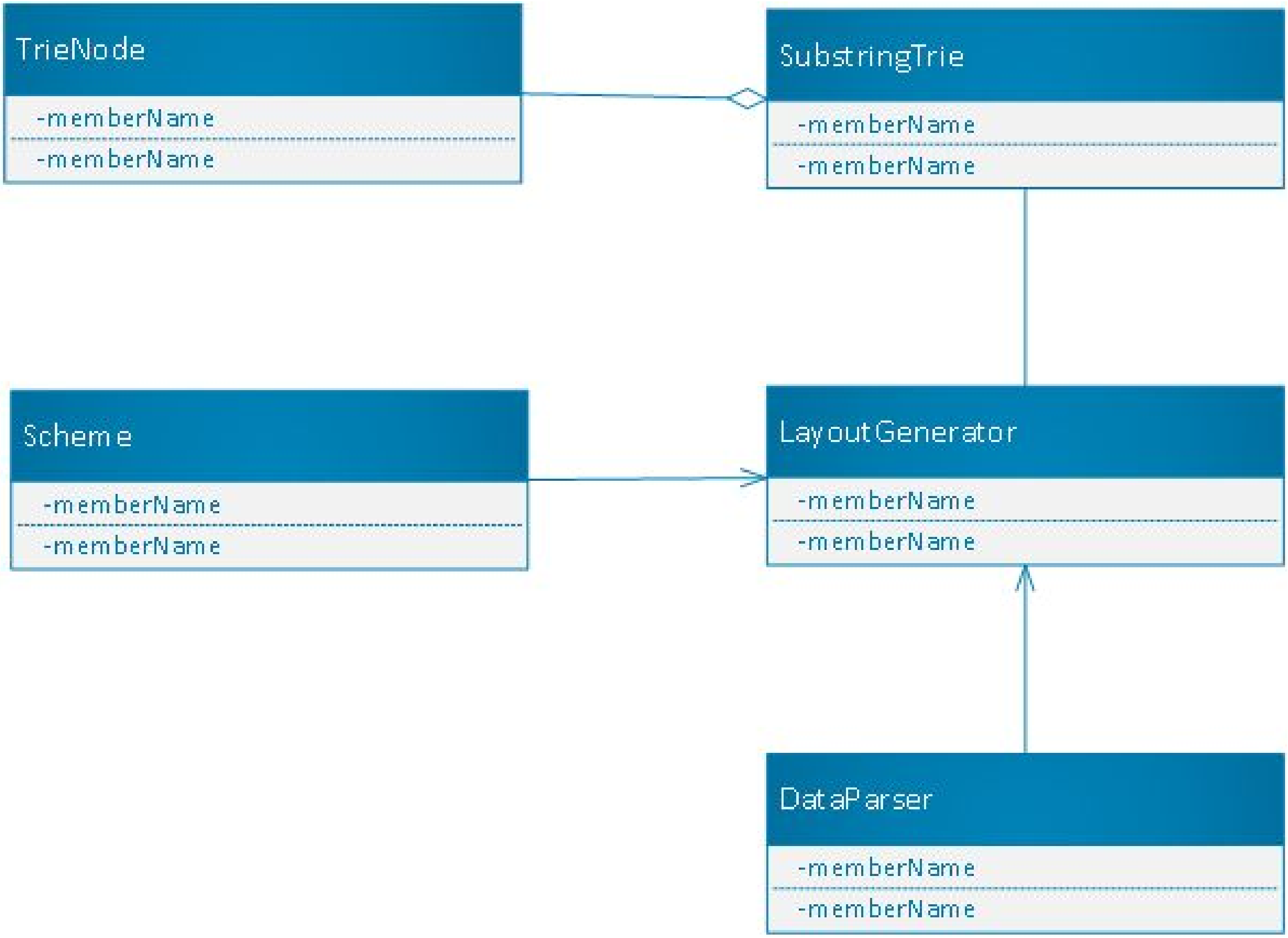
Deployment Diagram



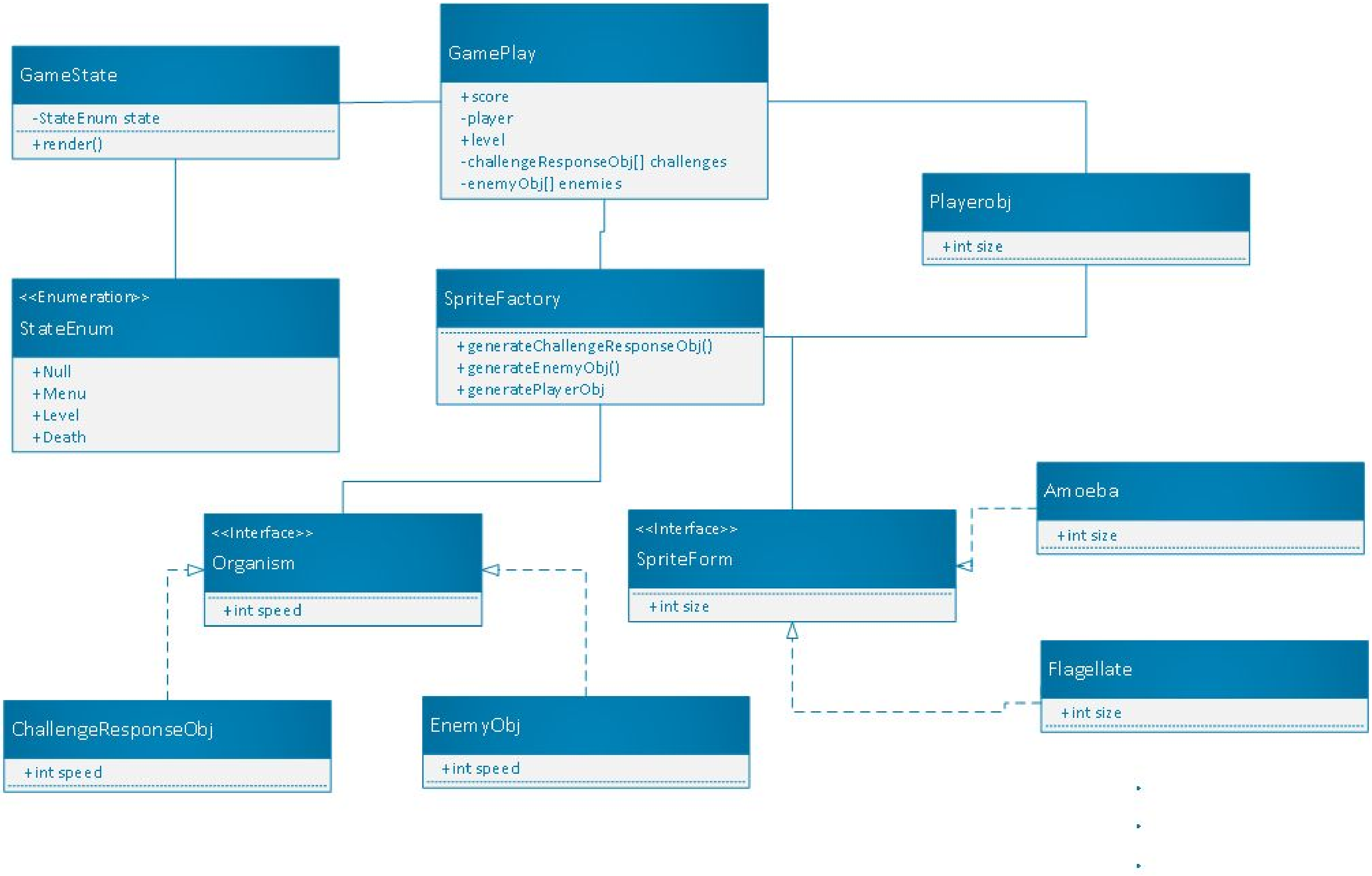
Interpreter - Class diagram



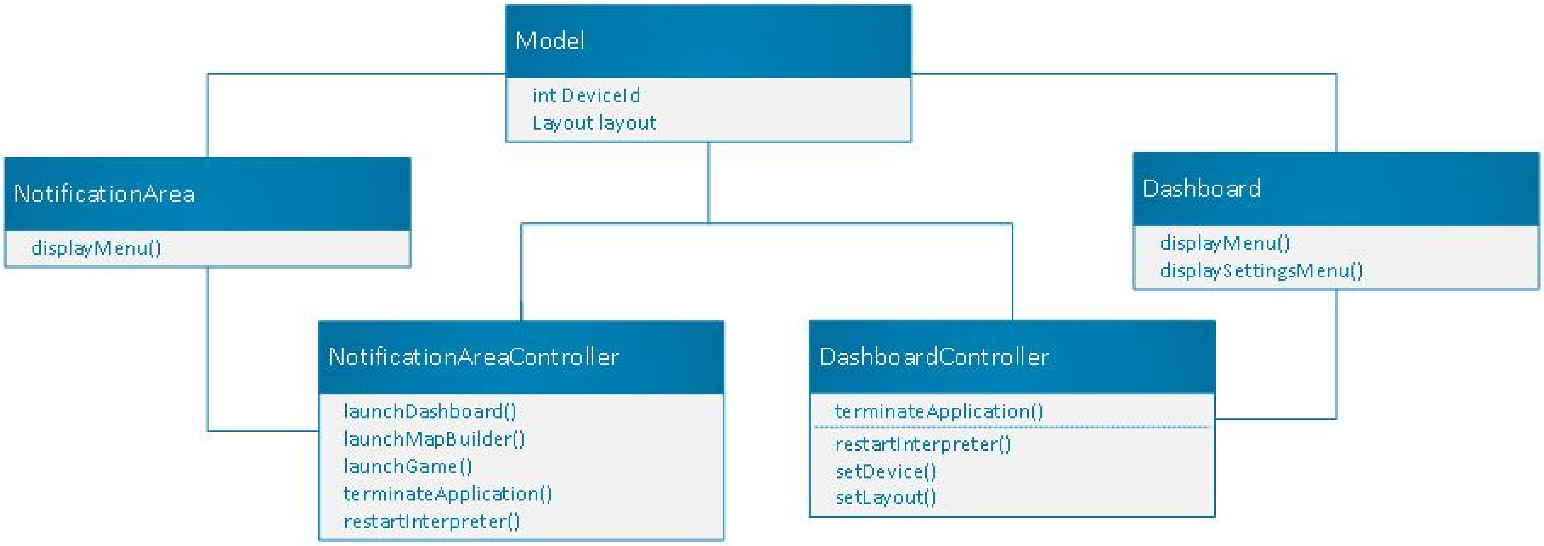
MapBuilder - Class Diagram



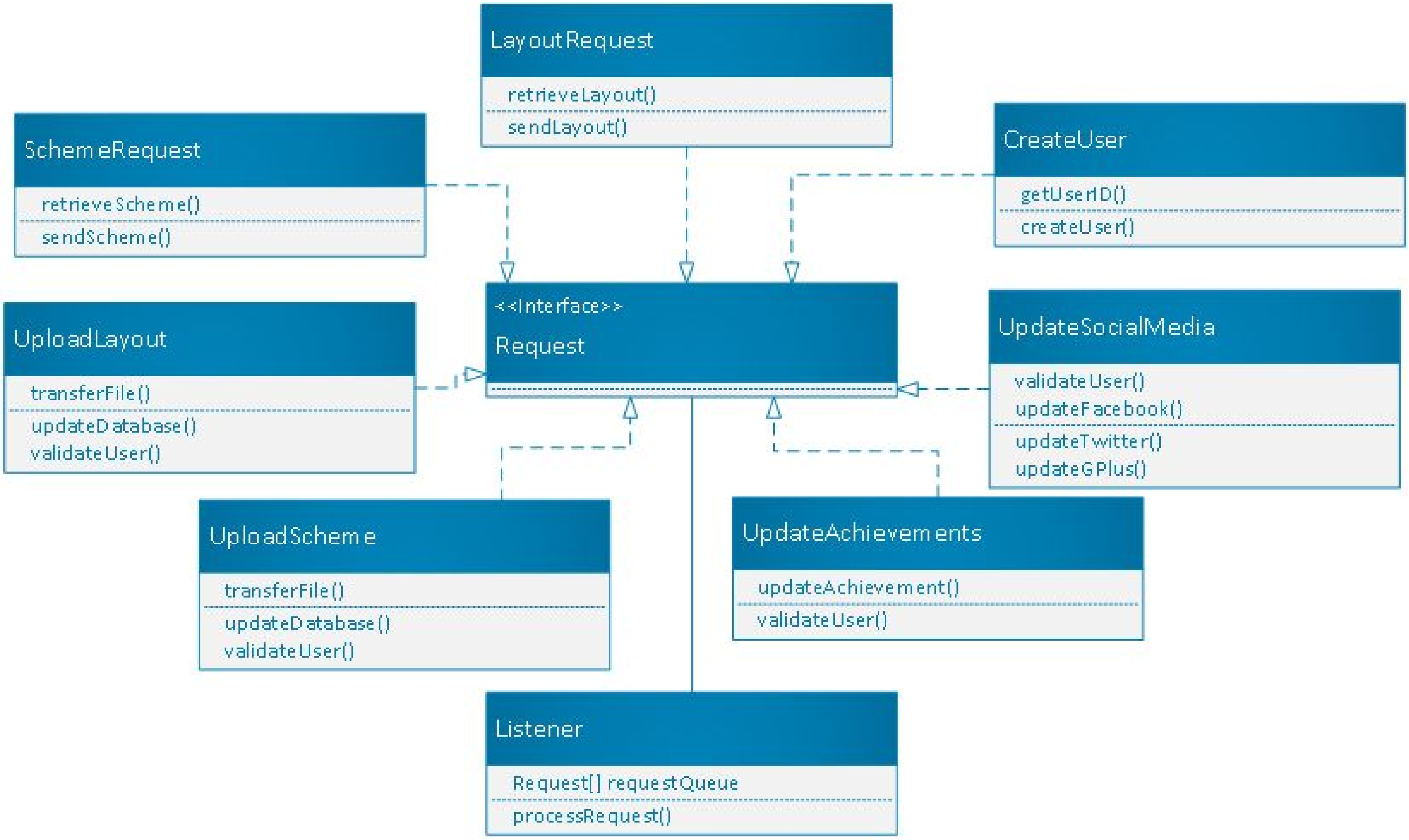
Game - Class Diagram



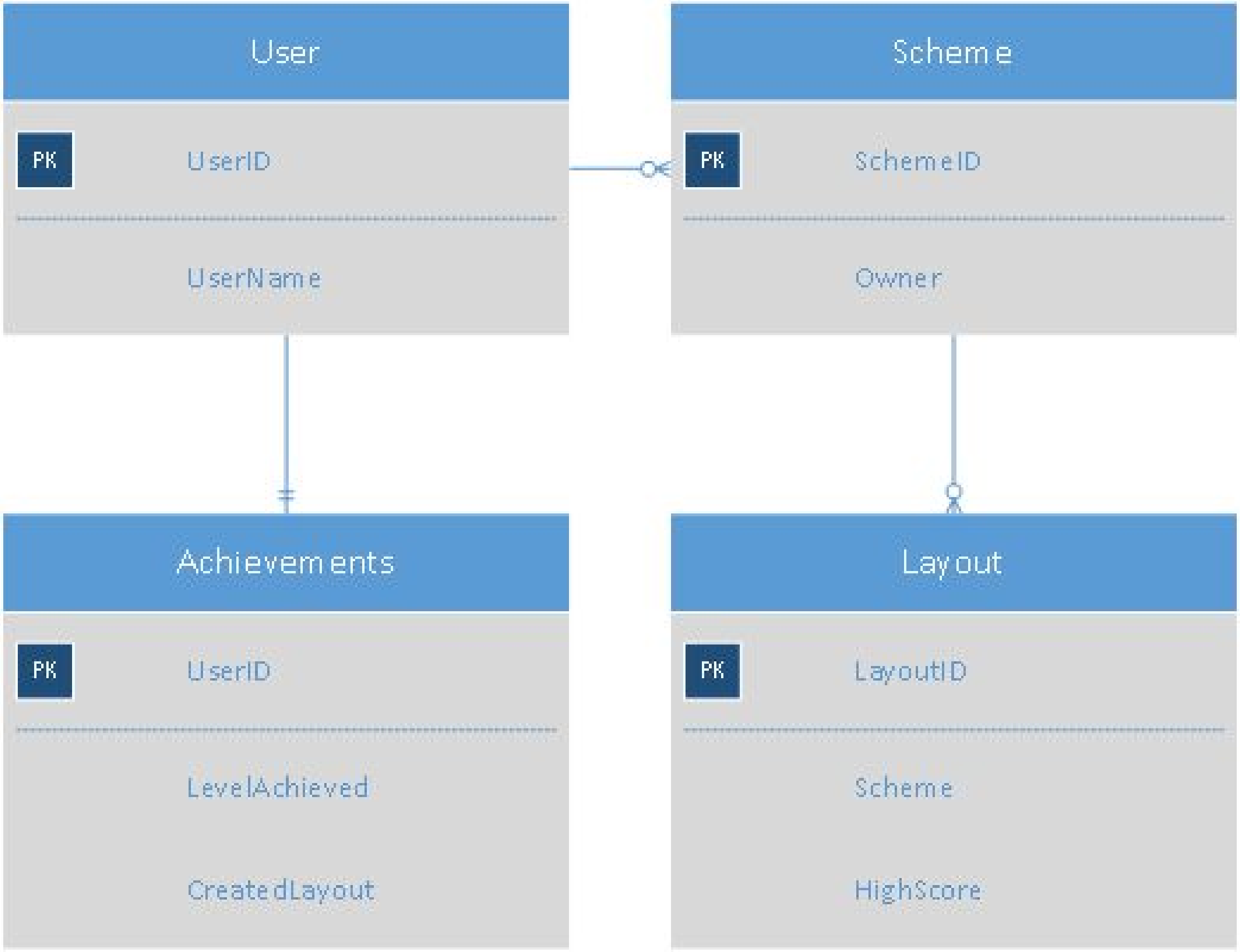
Controller - Class Diagram



Server - Class Diagram



Server - Entity Relationship Diagram



Appendices

Source code available at https://github.com/SamTheWizer/ChopChop