Student Lap Tracker

Architecture Documentation

Patrick Oyarzun, Joe DeHart
Vaishali Patel, Duke Meche, Robert Knott

ABSTRACT
This document describes the architectural decisions and practices used in the creation of our Lap Tracking project. The Lap Tracker uses an MVC model, and will use a web application in order to interact with the user, and that application will communicate with a controller to ensure that application invariants are maintained. We will use the Python runtime to host a JSON interface on Google App Engine's platform-as-a-service (PAAS).

TABLE OF CONTENTS

LIST OF FIGURES

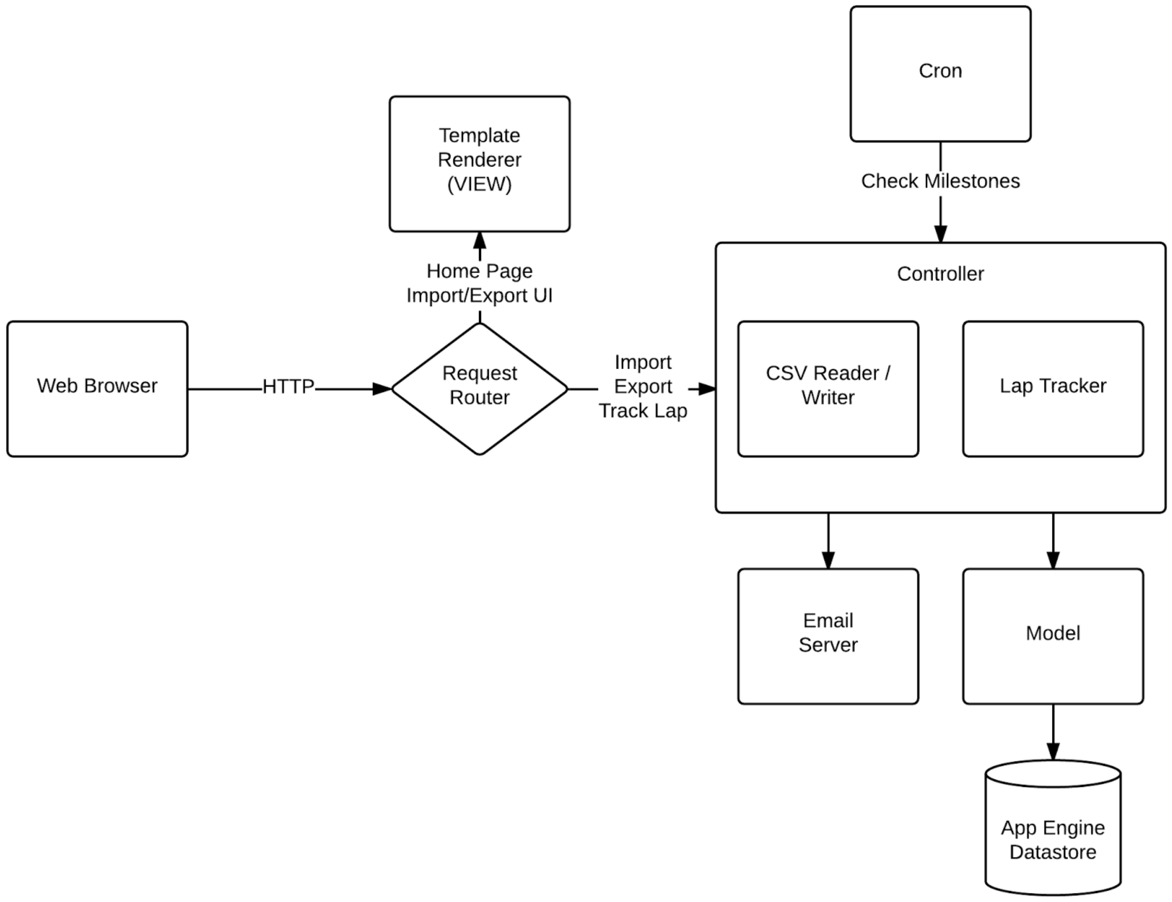
INTRODUCTION
The System Architecture Document covers the architectural specifications used to represent the product and a complete system model that shows the different components in a system, their interfaces and their connections.

This document provides a comprehensive architectural overview of the system, using Model View Controller architectural view to depict an aspect of the system.

ARCHITECTURAL STYLE(S) USED
The application will be implemented using the Model View Controller (MVC) style to coordinate implementation of features without introducing unwanted complexity into the system. The

primary concern of the application is managing and maintaining a dataset of students and their respective progress on a running track.

ARCHITECTURAL MODEL



TECHNOLOGY, SOFTWARE, AND HARDWARE USED

Technology

The server is hosted on Google App Engine under the Python runtime. App Engine is a platform-as-a-service(PAAS) solution designed to allow small teams to develop distributed applications with minimal work. Several components of the application use the services provided as part of the platform. The underlying database is the App Engine datastore, a highly scalable distributed database built upon a distributed storage system called Big Table. Our application will not likely require the scalability features provided by the datastore, but having them available with zero configuration increases the scope of use cases our application can grow to support in the future. The application also makes use of Google's distributed memcache architecture, the email service, and a message queue which allows maintenance tasks to run on a schedule, or outside the context of an HTTP request.

Software and Hardware Requirements

The required software includes: The App Engine Python sdk, a web browser, and the developer's preferred environment for developing python applications, e.g. Sublime Text 2. In order to receive email reports, an email client is required.
Hardware requirements include: A barcode scanner with the ability to emulate a standard keyboard, a web-connected machine with a usb port and support for keyboard use.

Communication

The application server communicates with the datastore through protocol buffer RPC requests. This is handled transparently by the AppEngine sdk. Client server communication is accomplished through a JSON REST interface. The client is implemented in javascript, so a JSON interface is ideal in order to minimize the development time overhead associated with communication.

RATIONALE FOR YOUR ARCHITECTURAL STYLE AND MODEL

The primary user interface is implemented through a web browser, and MVC allows the application logic to remain straightforward in terms of separating concerns and ensuring the application's invariants are maintained. MVC is a natural fit for web applications due to the inherent decoupling between frontend and backend logic that is imposed by the system.
Google App Engine was chosen due to the fact that, for small request loads, the service is entirely free. Our focus is on delivering a quality product and using App Engine allows us to focus on the business logic without being slowed down by system-level concerns which are outside the scope of the project. For example, load balancing, server restarts, and routine server maintenance are all handled seamlessly by Google while providing near-zero downtime for our application.

EVIDENCE THE DOCUMENT HAS BEEN PLACED UNDER CONFIGURATION MANAGEMENT