32386120 | Alec Ng
32136111 | Justin Lane
45909116 | Aviral Garg
31580129 | Stephen Hu
90294133 | Yue (Mark) Chen
15991152 | Anh Duc (Andrew) Bui
54952130 | Tushar Kalra

CodePal

Design Document

INTRODUCTION

We plan to build CodePal as a web-based service that allows developers to quickly compile and execute small pieces of code without having to set up any working environments. CodePal supports various popular programming languages and lets users share their code snippets with one another. We also integrate different resources from StackOverflow, Repl.it and YouTube to assist the user's work.

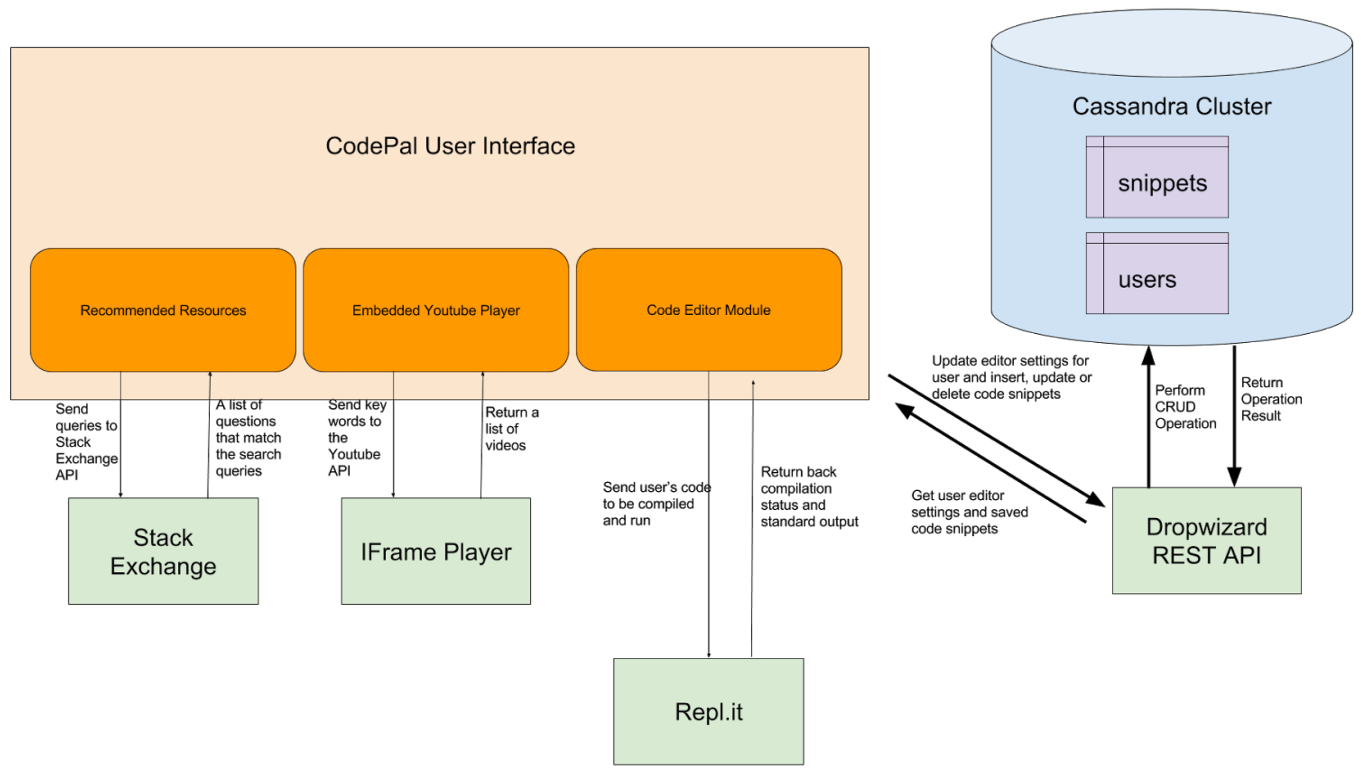
This document will explain the architecture and design of the application.

APPLICATION ARCHITECTURE AND RATIONALE

The application consists of the following major components:

1. Web application connected to a database via REST API for CRUD operations to handle user login authentication, saved code editor configurations and stored code snippets.
2. Repl.it API to generate a code editor module.
3. IFrame Player API which lets embed a YouTube video player on the application.
4. Stack Exchange API that sends a list of questions from StackOverflow that are associated with the user's search queries.
5. GoldenLayout framework for UI styling

Dynamic View and Description


CodePal User Interface
Recommended Resources
Embedded Youtube Player
Code Editor Module
Cassandra Cluster
snippets
users
Send queries to Stack Exchange API
A list of questions that match the search queries
Send key words to the Youtube API
Return a list of videos
Send user's code to be compiled and run
Return back compilation status and standard output
Update editor settings for user and insert, update or delete code snippets
Get user editor settings and saved code snippets
Perform CRUD Operation
Return Operation Result
Stack Exchange
IFrame Player
Repl.it
Dropwizard REST API

Dropwizard RESTful API

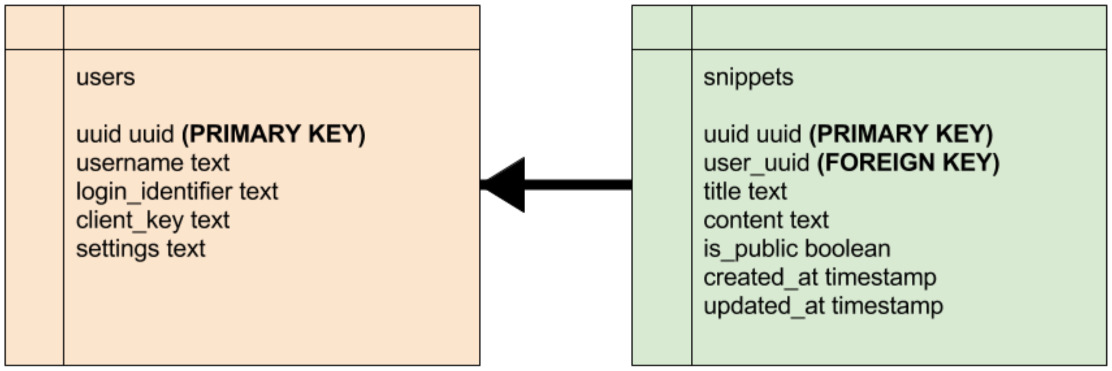
CodePal's backend service handling user management and enabling create, read, update, and delete operations for snippets is powered by the Dropwizard web framework. This service runs on an EC2 instance with high availability and handles up to 100 requests per second with the ability to scale up based on traffic volume.

This API provides a "/snippets" endpoint for creating, reading, updating, or deleting a snippet through POST, GET, PUT, and POST HTTP requests, respectively. The user's code editor settings are updated through a PUT call to the "/settings" endpoint.

All calls to the API will be authenticated with basic HTTP credentials required in the header, which will be encrypted in transit. User authorization will be enforced by checking the ownership of a specified content (e.g. User A should not be able to view User B's snippet if it is private).

In managing the Cassandra cluster and session connection in the Dropwizard application, it was handled as a singleton object in order to minimize the overhead of managing calls to the Cassandra cluster. In the lifetime of the application, only one connection instance is initialized and persists to be reused for all database operations.

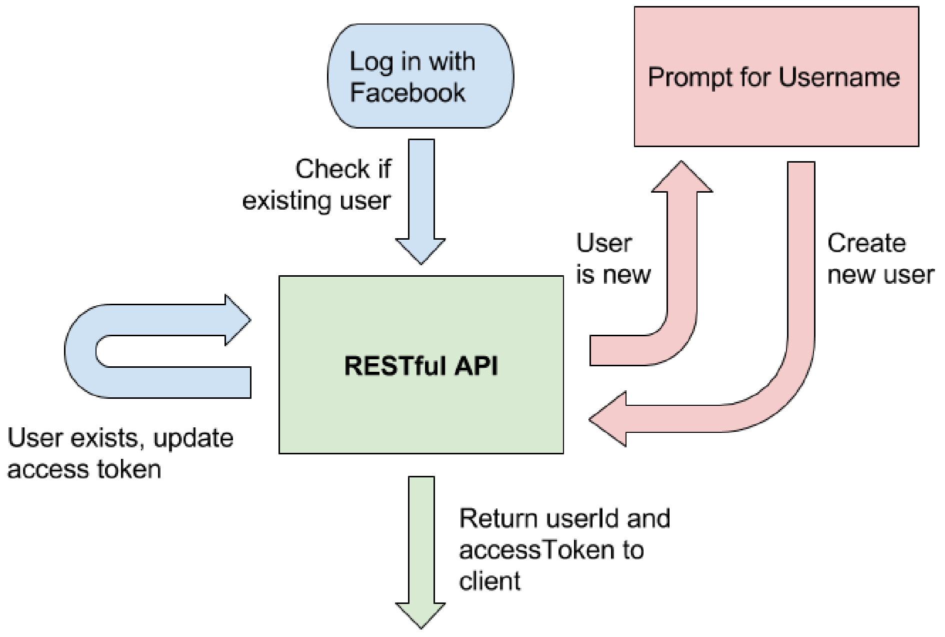
Database Model



CodePal uses Cassandra clusters running on EC2 instances for its data store. Cassandra is a distributed NoSQL database with optimized replication settings, which will enable high availability and partition tolerance. Thus, this will ensure CodePal is able to scale horizontally with the growing number of snippets being created and additional users registering for the service. Also, the data is replicated to eliminate a single point of failure and allow the data to be accessed with high availability even in the case of an outage on a cluster.

The database schema is simple, currently with just a single table for users, and another for snippets. Each snippet belongs to a single user (the creator), and is referenced by its foreign key *user_uuid.* The user is able to determine if a snippet they have created should be public or not. A public snippet will be viewable by anyone with the link, whereas a private snippet is only viewable by its creator.

When a user registers for CodePal by signing in through Facebook, their user record is entered into the users table and their login identifier is saved, as well as their client key. This information will be encrypted and sent with every request through the API, which authenticates their request.

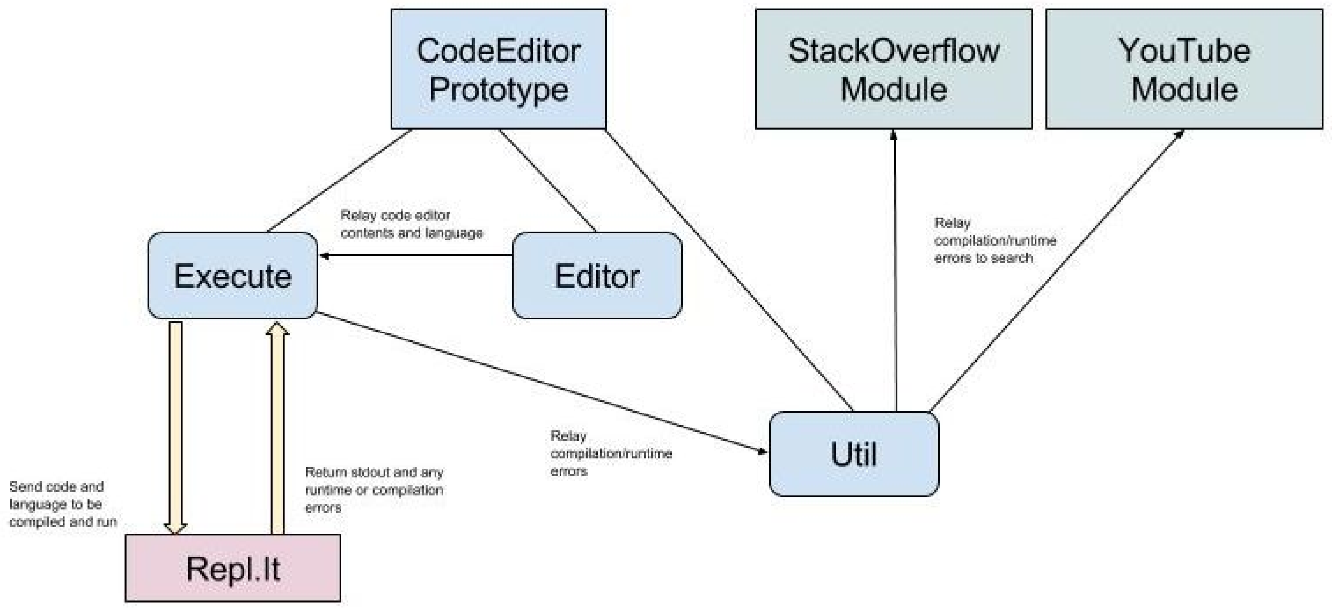


Login Flow Sequence Diagram

Repl.it

Repl.it is a competitive coding platform. It exposes an API that allows the compilation and running of raw source code within a set of languages they support. Given a supported language, source code and a client secret key and access token that is obtained by registering an account with their service, Repl.it will return a response that will contain any compilation errors and if the code is run, the and any standard output generated from the code. The platform limits registered apps to a maximum of 25000 requests to the *execute* endpoint. For the purposes of this assignment, this limit request is more than necessary but for a scaled commercial application, they offer commercial licenses with several options for payment depending on scale.

This component uses a prototype design pattern to encapsulate different subcomponents, such as the *execute module* which handles callouts to the Repl.it module and the *editor* module, which handles the code editor plugin. Each submodule is built using the revealing module pattern, in which an object literal encompasses public functions and properties that can be used by other submodules. Visualized in the diagram below, the *execute* gets the code editor content and language to be sent to the Repl.it API to be evaluated. The response from the API is outputted by the *execute* module to be displayed by the user. Any errors are passed to the *util* modules which “talks” to the StackOverflow and YouTube modules, in order to independently search the error.


CodeEditor Prototype
StackOverflow Module
YouTube Module
Relay code editor contents and language
Execute
Editor
Relay compilation/runtime errors to search
Relay compilation/runtime errors
Util
Send code and language to be compiled and run
Return stdout and any runtime or compilation errors
Repl.It

YouTube

Codepal uses the YouTube API to provide a search functionality and an embedded video player. This would enable a streamlined environment for development and simultaneous video playback. YouTube's API key is easily obtained by a Google account and registering the application. YouTube's data API offers search implementation that returns the information of a video (such as channel or playlist) that matches the specified search parameters. Given a search input, the return value can be in the form of a list in JSON format containing the information of the video as the fields. This will be used in conjunction with the YouTube API's iframe implementation that allows for embedding of the YouTube video player in independent websites and applications. An iframe tag is generated with the correct YouTube URL that can be added to another project to enable an embedded YouTube video player.

YouTube component uses Model-View-Controller design pattern. It allows encapsulation of various subcomponents, such as ***Model*** that contains all the functions that directly deal with the data used for the YouTube component and the ***View*** that contains the functions that render the visible elements of the YouTube Component. Model and View functions never interact with each other so the Controller initializes them and facilitates those interactions. Controller interacts with Model to store data received from the YouTube API response in the browser's localStorage and communicates with View to provide the list of videos stored. View also adds event listeners to the YouTube's HTML elements.

Model

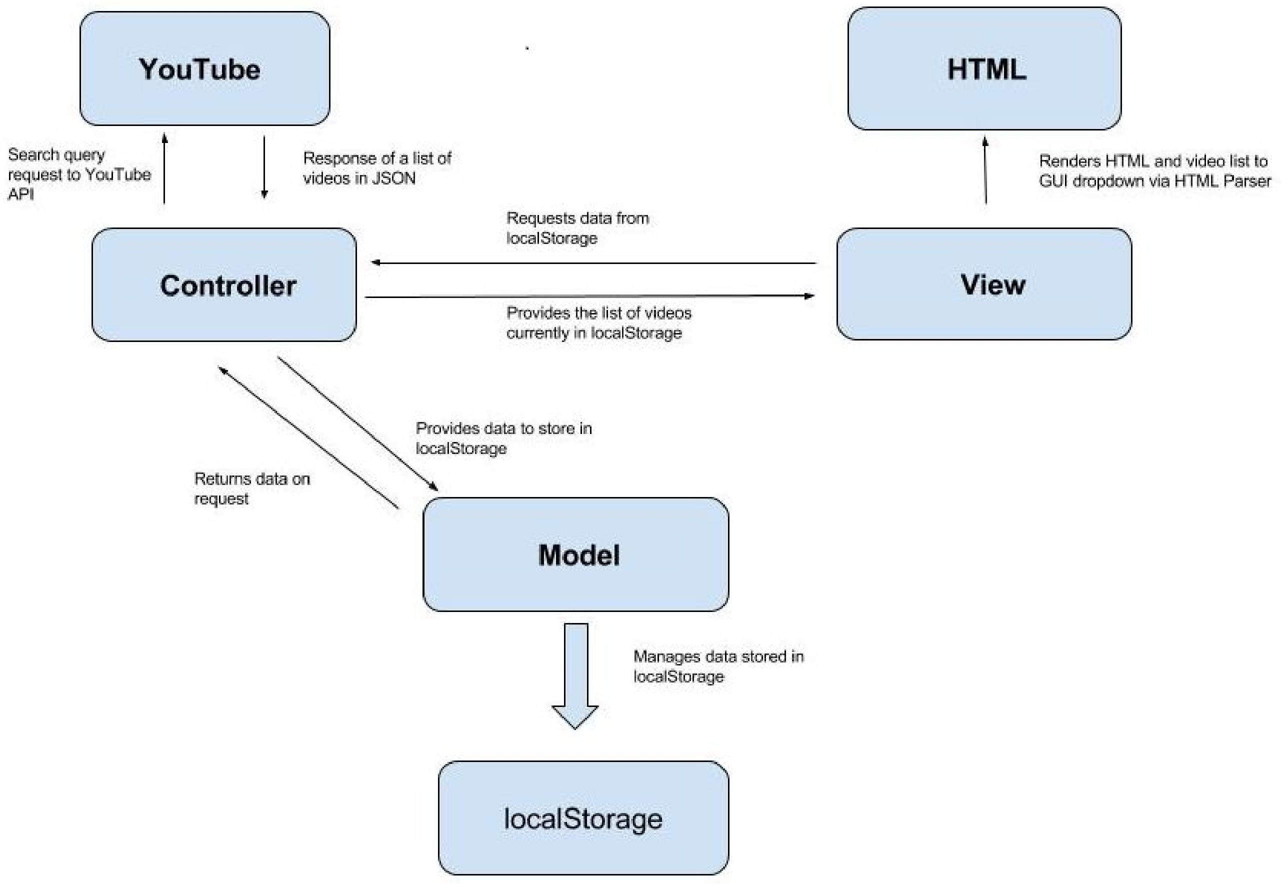
- Initializes localStorage for storing
- Provides getters and setters for localStorage data

Controller

- Provides View with the information of the list of videos currently in localStorage
- Provides Model with list of videos (JSON format) to store in localStorage

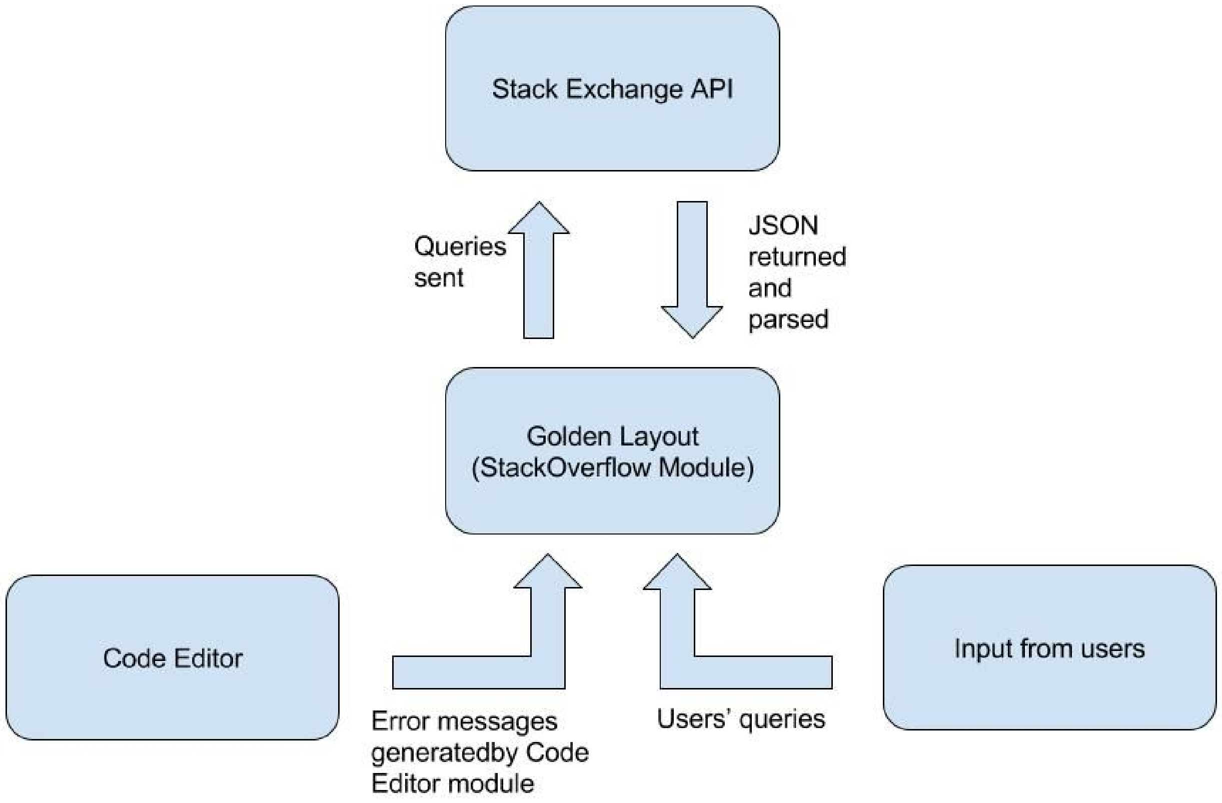
View

- Initializes YouTube's HTML components
- Add event listeners to these components (Search Button)
- Renders the YouTube component


YouTube
HTML
Search query request to YouTube API
Response of a list of videos in JSON
Renders HTML and video list to GUI dropdown via HTML Parser
Requests data from localStorage
Controller
View
Provides the list of videos currently in localStorage
Provides data to store in localStorage
Returns data on request
Model
Manages data stored in localStorage
localStorage

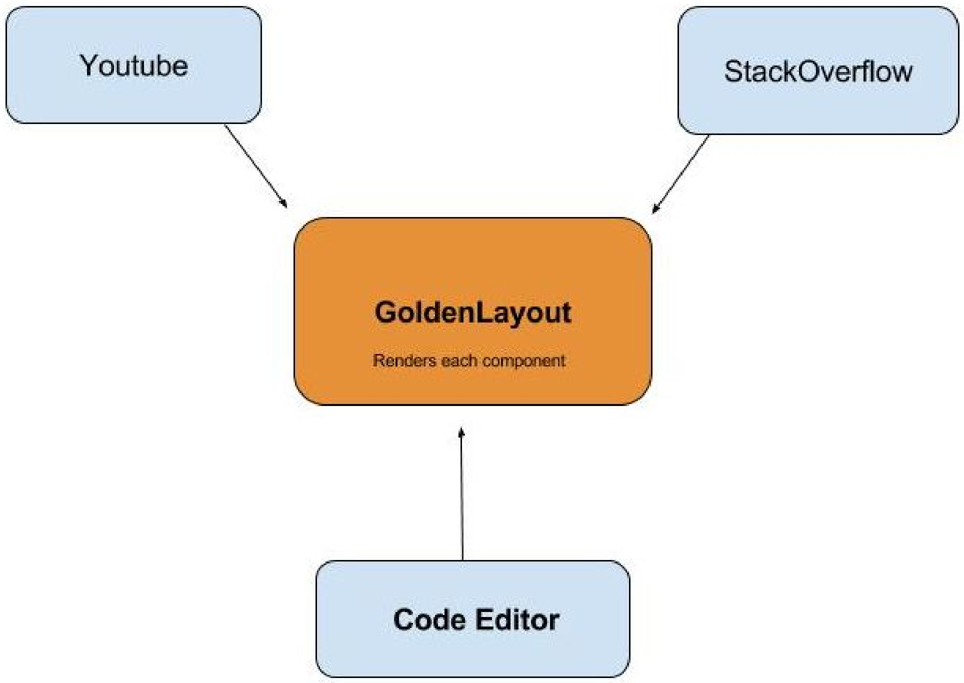
StackOverflow

StackOverflow is an online community for programmers that branches off of Stack Exchange. It features questions and answers on a wide range of topics in computer programming. The Stack Exchange API provides search functionality for questions and answers using specified parameters. Given the search query which can be either passed by the Code Editor module or entered by the user, Stack Exchange will return a list of ids that are corresponding to the questions that match the search query, along with other information in JSON format. Using those question ids, we are able to send GET requests again to Stack Exchange in order to obtain the bodies of its accepted answers.

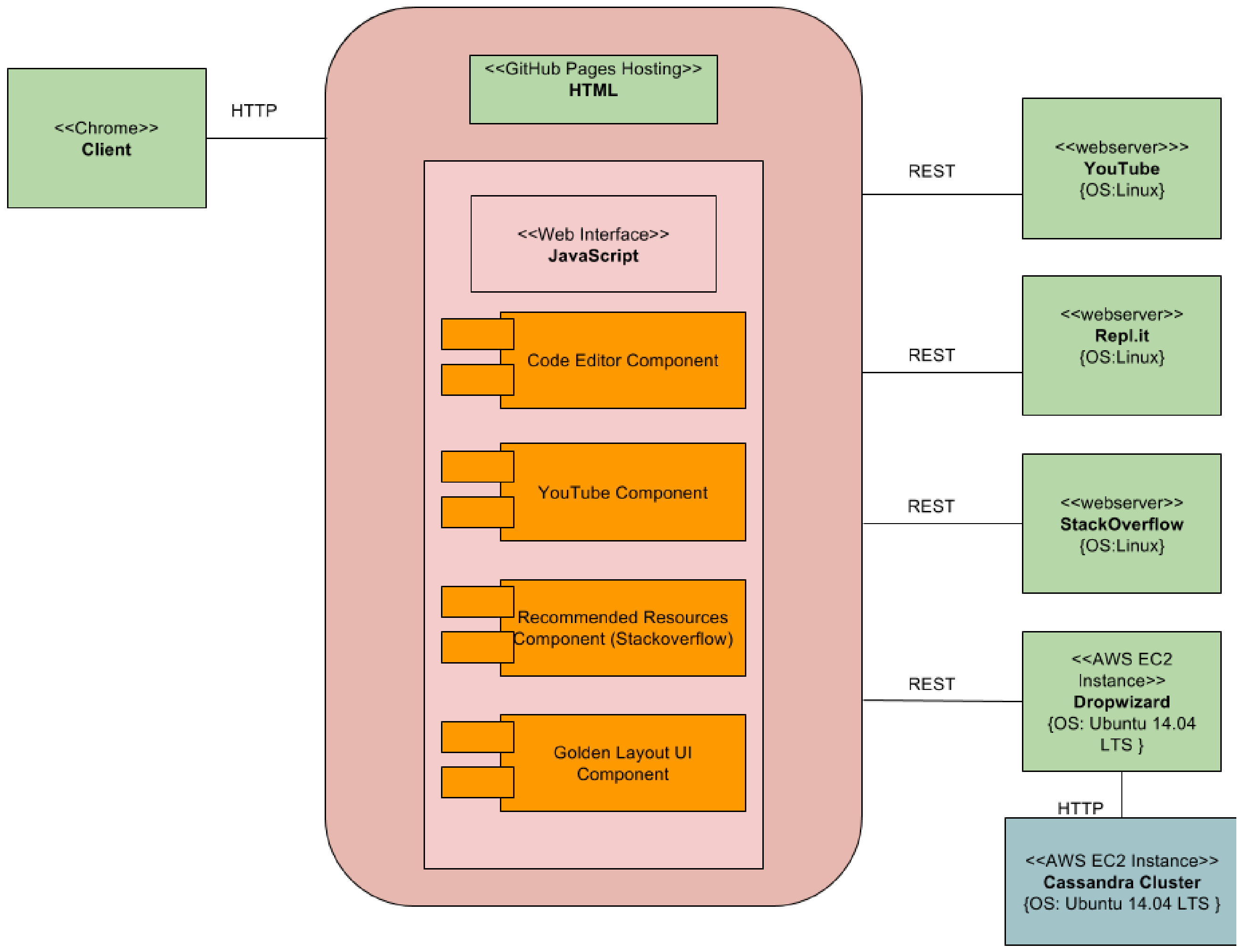


GoldenLayout

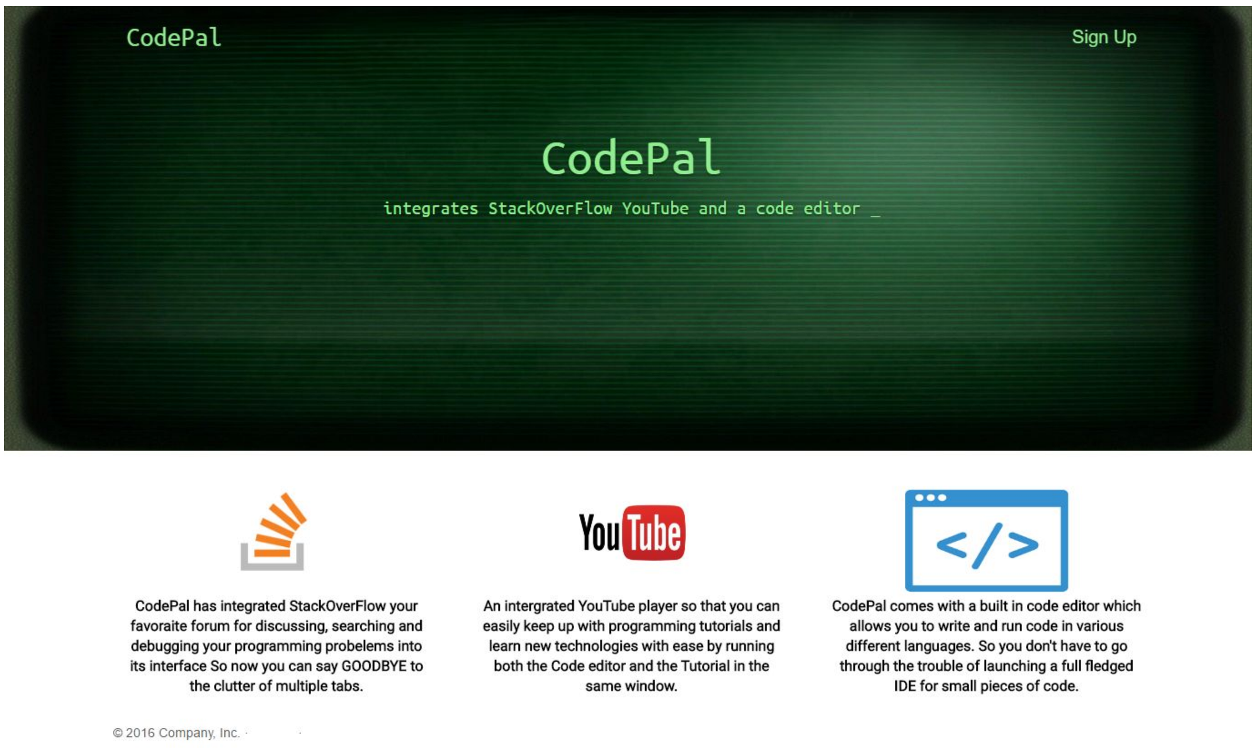
GoldenLayout is a layout manager framework that supports browser based multi-window/multi screen webapps. GoldenLayout provides a fully customizable template in the form of individual components that can be altered for each specific user's needs (resizing, minimizing, drag and drop) through built in default functions. Each of the CodePal modules will be contained inside separate GoldenLayout components. It allows configuration and instantiation of the layout, registration of the components to specify what each component is (code editor/youtube player/stackOverflow), and lastly the initialization of the layout to display the content of the entire page with each of the modular components/blocks.



Deployment Diagram - Static View



Landing Page



CodePal has integrated StackOverFlow your favoraite forum for discussing, searching and debugging your programming probelems into its interface So now you can say GOODBYE to the clutter of multiple tabs.

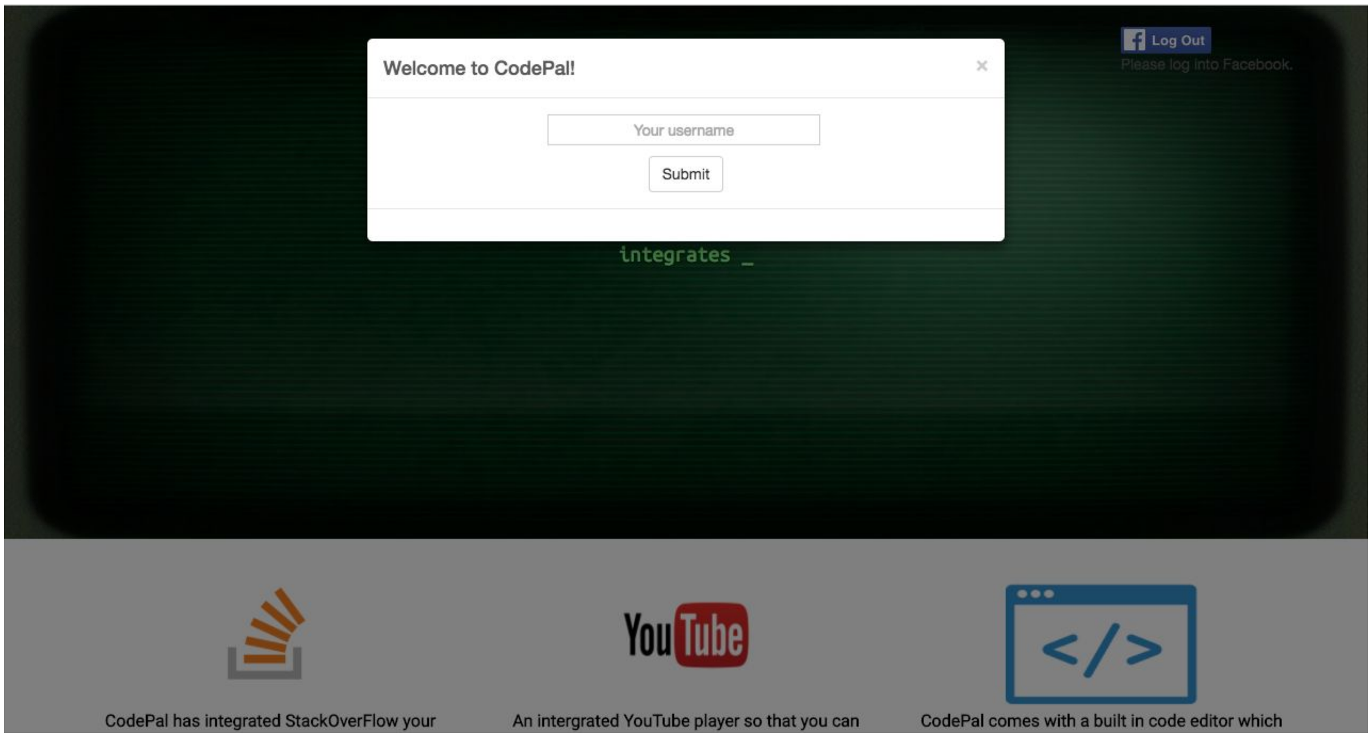
An intergrated YouTube player so that you can easily keep up with programming tutorials and learn new technologies with ease by running both the Code editor and the Tutorial in the same window.

CodePal comes with a built in code editor which allows you to write and run code in various different languages. So you don't have to go through the trouble of launching a full fledged IDE for small pieces of code.

© 2016 Company, Inc. ·

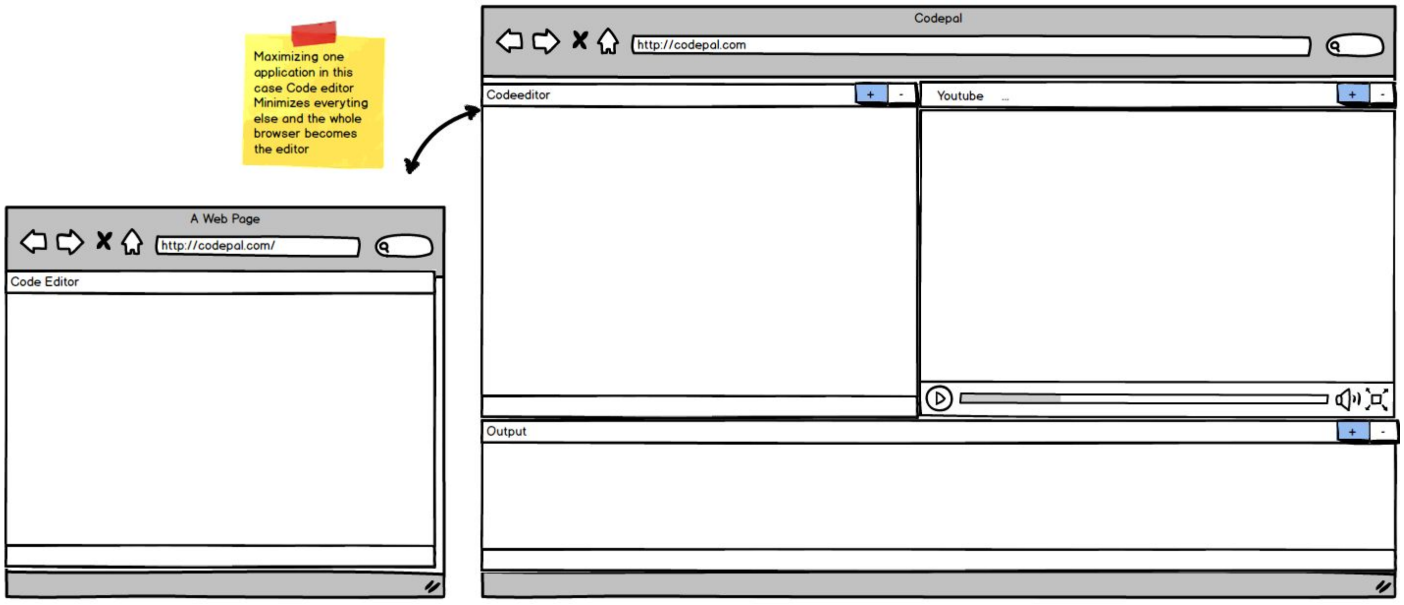
We decided to make the landing page very simple and clean as it goes along with the theme of our project. We used Typer.js (http://steven.codes/typerjs/docs/index.html) to print animated text about our application. Then we added logos and writeups about the core features of our application following along their respective branding agreements for legal use.

Login Page


Welcome to CodePal!
×
Your username
Submit
Log Out
Please log into Facebook.
integrates _
CodePal has integrated StackOverFlow your
An intergrated YouTube player so that you can
CodePal comes with a built in code editor which

Upon a login from the user, we will send POST requests to the server to check if the user already exists in the database. If not, which means that the user has not used CodePal before, we will prompt the user to enter a username in order to register his/her account with CodePal using Facebook User ID and Access Token. For future logins, we will just need to update the new user’s Access Token in the database.

Basic UI Mockups


Maximizing one application in this case Code editor Minimizes everyting else and the whole browser becomes the editor
A Web Page
http://codepal.com/
Code Editor
Codepal
http://codepal.com
Codeeditor
Youtube
Output

Validation

UI validation is done on a biweekly schedule where the design and selected theme is presented to users for approval. By incorporating the suggested changes from the feedback, the user interface is continuously refined for a more easy to use/user-friendly environment. The design process also includes changing the layout and functionality based on the changing requirements requested by the client.