Software Requirements Specification

for

CodeX

Version 1.0 approved

Prepared by Karthik

Vasavi College of Engineering

5-10-2020

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
CodeX	5-10-2020	Primary Edition	1.0

1. Introduction

1.1 Purpose

*The purpose of the Online Coding Platform (tentatively named as **CodeX**) is to provide an environment which helps to ease the learning process associated with programming abilities. The interface after authentication provides a convenient editor where we can create, edit, compile and execute our program.*

1.2 Document Conventions

In general, this document follows the IEEE formatting requirements. font size 12 is used throughout the document for text. Document text is single spaced. No special formatting techniques are used.

1.3 Intended Audience and Reading Suggestions

The project done is intended to apply the concepts learned in web programming and services(WPS) and Data Base Management System(DBMS), to develop a Full Stack Web application and it is basically a prototype of a conventional Online coding platform, This has been implemented under the guidance of college professors.

1.4 Product Scope

The Product we are developing is basically to provide a learning environment that consolidates students programming abilities and learning important algorithms by applying it on practical problems. This helps making students adept in competitive coding as well. The salient features include a convenient editor. Conduction of weekly coding competitions and a weekly leaderboard that helps in tracking a person's progress.

1.5 References

https://www.npmjs.com/, https://reactstrap.github.io/, https://material-ui.com/ these were the resources mostly referred to make user interfaces which match current trends.

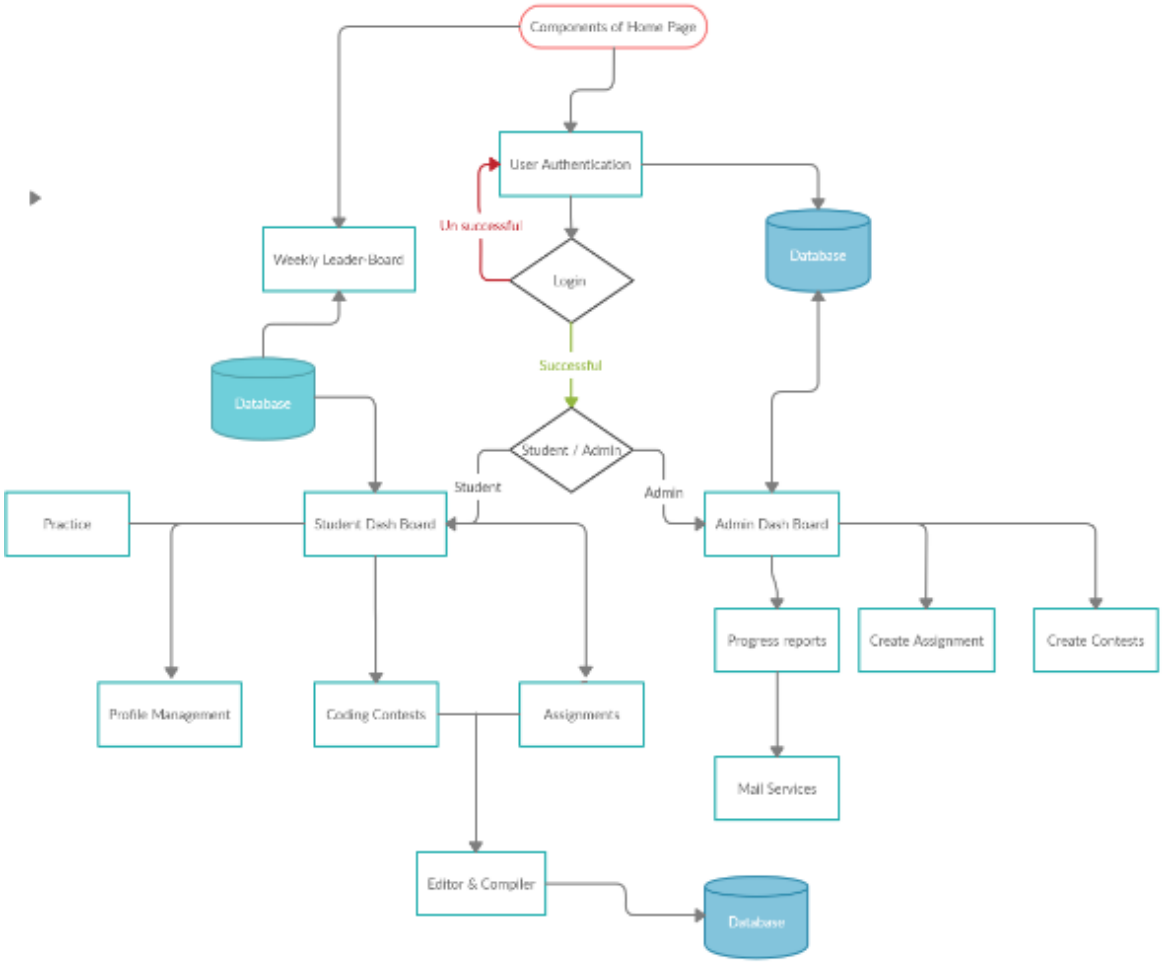
2. Overall Description

2.1 Product Perspective

The website home page consists of few components

- *User authentication portal*
- *Weekly leaderboard*
- *Dashboard (Student / Admin)*
- *Editor to code, compile and execute codes*
- *Participate and Conduct contests*
- *Snippets for good user experience*

2.2 Product Functions



2.3 User Classes and Characteristics

Users of our web application could extensively practice, compile and execute code. Our Application basically support two types of user privileges student and admin. The student

mode basically helps to practice and submit codes with our compiler (Editor) for coding contest happening and assignments given by there faculty. Admin (Faculty) can basically create the assignments or contests to students with a ease. And administrate overall performance of students with reports and stats provided by our application.

Functions of Student:

1. *Dashboard showing their performance*
2. *Weekly leader board*
3. *Editor to code, compile and execute*
4. *Support to submit Assignments*
5. *Sophisticated coding environment features*

Functions of Admin:

1. *Create coding contests*
2. *Create assignments or exams*
3. *Monitor performance of students*
4. *Send results to students (mail services)*
5. *Access to codes of student*

2.4 Operating Environment

Operating environment is as follows:

- *Operating System: Linux, Mac, Windows*
- *Database: mongo DB*
- *Platform: Django/ React*
- *Client/Server Model*

2.5 Design and Implementation Constraints

Our Website basically runs 3 tire architecture where front end basically runs on one port and backend runs on other port. And they communicate using restful API's, so we require servers which handle this. But to have performance we had to prefer some cloud for deployment. Since our model requires to deal with great amount of codes or files it must store in BLOBs or Bason Objects in our data base. So, we had to shift to No SQL database. Since mongo suits best for purpose.

2.6 Assumptions and Dependencies

We are depending on Hacker Earth API for compiling which is basically free as surveyed on 2nd oct, 2020. This could keep the pricing later but there are many potential API' s available which can be used. We are depending on cloud-based server which are mostly available but when this product undergoes changes and surpasses prototyping stage needs commercial requirements to have Virtual machines in cloud to have better performance with huge traffic. But could be sorted we leave few areas web app to ads could meet its own requirements. And we require a credit card to avail necessary credits in clouds to have demo runs.

3. External Interface Requirements

3.1 User Interfaces

1. *Front-end: React. Js, HTML, JavaScript, CSS, Bootstrap*
2. *Backend: Django, mongo DB*

3.2 Hardware Interfaces

1. *Windows/ Linux / Mac*
2. *A browser which support fetch and restful calls*

3.3 Software Interfaces

Following are software used:

1. *Operating System: (Linux/ Mac/ Windows) Any operating virtual machine can work for the purpose of serving the purpose.*
2. *Database: (Mongo DB) No-SQL database serves the best purpose to store data which mostly has files or text.*
3. *React: Easy to create and can load real time data with a ease as it has lifecycle methods to make our task easy.*
4. *Bootstrap: It has grid layout and its components are responsive. So, using them removes pain of thinking over responsive to solve logic.*
5. *Django: it will run on server and our product's business logic stays there.*

3.4 Communications Interfaces

This web app is supported by all web-bowsers. You can use our editor or upload files to compile and execute codes.

4. System Features

4.1 Weekly Leaderboard

4.1.1 Description and Priority

Leaderboard is important feature where it establishes a spirit of learning among fellow mates with a small sense of competition.

4.1.2 Stimulus/Response Sequences

- *Put custom filters to search and get positions on leaderboard.*
- *Enlists the users with their accordance of rank*

4.1.3 Functional Requirements

This feature will provide will good looking list where we can find about the performance of students. All users can custom filtered to get required details but when we do not have any users to fetch it will result in sorry page.

REQ-1: Django at server side running
REQ-2: restful API calls
REQ-3: Mongo DB

4.2 Student Dashboard

4.2.1 Description and Priority

It will be providing the user with appropriate notifications, tasks upcoming, assignments and coding contests. This would keep user from missing out important notification when user is lost of track of time.

4.2.2 Stimulus/Response Sequences

- *Notifications showing all necessary details*
- *Tracks his abilities, assignments and contests, participated by user and also upcoming ones too.*
- *He can submit assignments which faculty created with a ease and without tension of mailing or submitting.*

4.2.3 Functional Requirements

This feature lets us track all the records of our coding.

REQ-1: Django at server side running
REQ-2: restful API calls
REQ-3: Mongo DB

4.3 Admin Dashboard

4.3.1 Description and Priority

This will let admin to create thoughtful challenges and assignments without restricting themselves in their bounds. It also let Admin to monitor and evaluate performance of individuals as well as whole too.

4.3.2 Stimulus/Response Sequences

- *Create assignments.*
- *Create coding contests*
- *Performance analytics*
- *Details reporting can be done which let faculty to focus on weakness of students.*
- *Mail services to post assignments and alert users.*

4.3.3 Functional Requirements

This feature lets us track all the records of our coding all individuals who under the admin.

REQ-1: Django at server side running
REQ-2: restful API calls
REQ-3: Mongo DB
REQ-4: STMP API's

4.4 Editor

4.4.1 Description and Priority

This is convenient editor which let us choose multiple languages and can help the users to maximize the throughput without struggling with configuration errors.

4.4.2 Stimulus/Response Sequences

- *Practice the code*
- *Run against testcases of the assignment and coding contests*

- *Ability to choose against wide range of languages*
- *Ability to choose against wide range of themes*

4.4.3 Functional Requirements

REQ-1: Django at server side running
REQ-2: restful API calls
REQ-3: Mongo DB
REQ-4: Hacker Earth API

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Codex website should be hosted in a cloud which supports most of latest self-load balancing tools like Kubernetics, Docker etc. These will help it stabilize the website from getting crashed due to huge traffic at times at contest.

5.2 Safety Requirements

Website has many advance features like auto code fillers and easy interface to create a contest. Regular check of log is important and always try to code changes to adapt to updating dependences.

5.3 Security Requirements

Website is most secured with password hashing and databases of personal details well maintained without the problem of privacy. Do not leak passwords or API keys by hosting your backend at unsecured domains and hosting sites. Mongo DB is used as cloud product, so just keep accounted connected with it safe.

5.4 Software Quality Attributes

It has best quality with features like modularity, combustibility, robustness, and easily understandable. The product is built with better scope of future and even we can avoid feature of API of Hacker earth, and even we can use scripts our own to run better.

5.5 Business Rules

Business is not part of our project. But to maintain it has ability to accept ads and also can provide source to host contest at lowest cost with function of charging cost to participants or even without charging.