Software Bot Design Proposal

Eric Fernandes, Sarah Wiessler, Miriam Podkolzin

Problem Statement

Crowdsourcing is a modern solution to challenges of a modern technical world. It allows companies to reach a wide variety of ideas at an extremely low cost, since they no longer are tied to the solutions of a couple salaried workers. However, issues arise when trying to make cohesive solutions from scattered ideas. We aim to address two closely related issues, that of low audience participation and the quality of solutions submitted, viewed through the frame of TopCoder, a software crowdsourcing platform.

TopCoder is a great platform to encompass real coding challenges, but as a freelance type forum it faces issues with priority. Many people *can* contribute, but when they have other jobs or life commitments to attend outside of the TopCoder challenges, these projects can fall to the wayside. When priority suffers, even when developers manage to devote time to solving an issue, since the feeling of responsibility is eroded in this low stakes environment, the solution achieved may not be practical, efficient, or effective. These issues of quality *and* quantity are intertwined, and we hope to contribute to a solution with our reward bot.

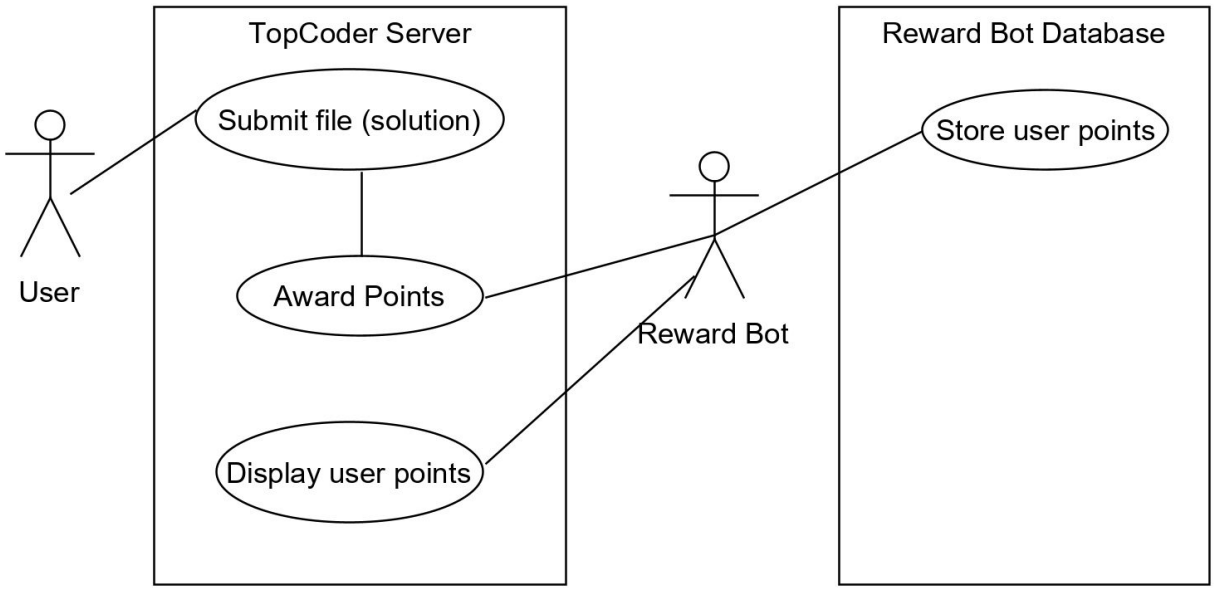
Bot Description

Our reward bot will reward community members for actions. The ultimate goal for our reward bot is to promote activity on TopCoder. It will do this by awarding points based on different actions from users. The bot will award less points for minor actions and more points for major actions. The bot will not have a conversation with users. Users will simply receive a notification of how many points they are receiving and for which action they are receiving points for. Thus the bot will just respond to events. The reward bot will also keep track of how many points each user has obtained in a day, then at the end of the day it will display a leaderboard of users with the top amount of points.

The reward bot is a good solution to the quality and quantity response problem within TopCoder because it will give users a sense of accomplishment. The increase of responses and the increase in quality of responses will ultimately lead to more activity on TopCoder which is also a plus. A bot being an agent of automation makes it a good solution to rewarding actions on TopCoder because it will be handling repetitive, automateted, predefined tasks. The point leaderboard the reward bot will post at the end of the day also adds more incentive to creating solutions of greater quality and adding more solutions. This feature of the bot additionally shows that it is a good solution. The tagline for our bot that captures the entire essence of our bot and project is: Your efforts, rewarded.

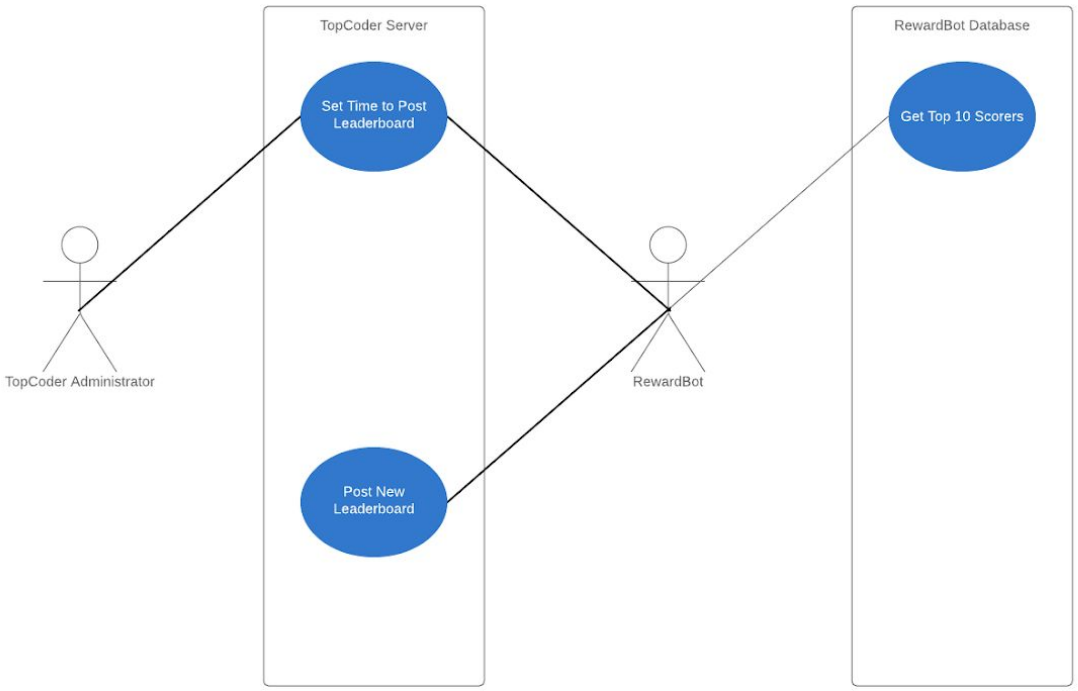
2 Use Cases

Submission Use Case



1. Preconditions: Users must be registered for the task and have a file to submit.
2. Main Flow: User submits solution, the reward bot awards points to the users profile for a plain submission, and displays updated point values. This action does not take into account the quality of the solution, just that one has been submitted.
3. Subflows: The interaction of the reward bot to the database is necessary for correctly accruing points. The reward bot needs to add the points for the submission while taking into account any points the user already has.
4. Alternative flows: If the user submits a corrupted file/incorrect file type, the reward bot will reject the file submission and not award any points.

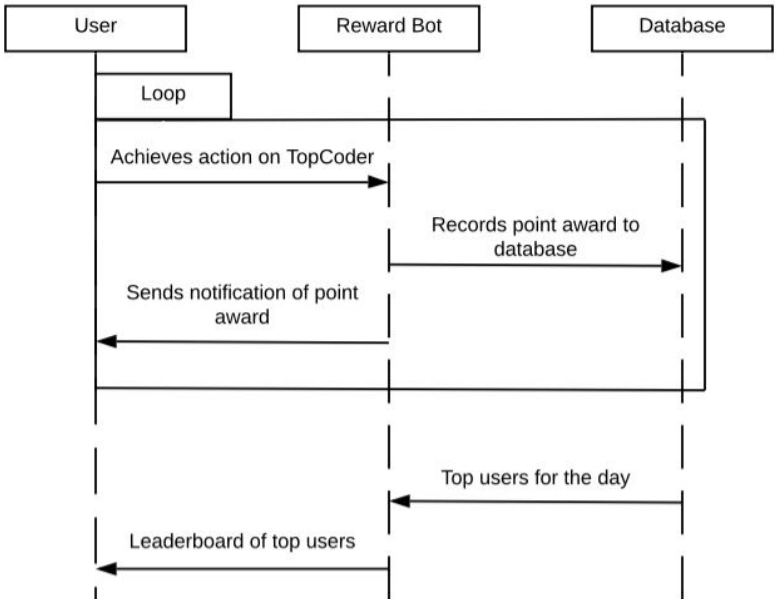
Leaderboard Use Case



1. Preconditions: RewardBot is connected to a leaderboard feed.
2. Main Flow: Administrators submit a time to the RewardBot for when it will update the leaderboard. The TopCoder server then publishes the leaderboard for the day.
3. Subflows: When the scheduled time comes the bot pulls the top 10 scorers from the database and submits the information to the feed.
4. Alternative flows: If there are less than 10 scorers RewardBot submits null accounts, which are displayed as empty fields on the website.

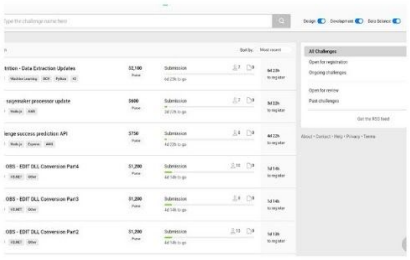
Design Sketches

Sequence Flow of Our Bot in Action



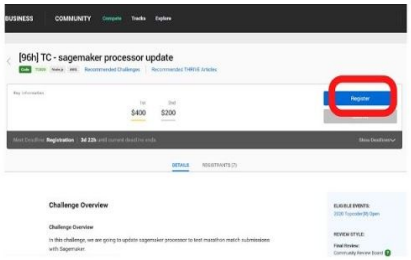
Primary Task Storyboard

BROWSE, REGISTER, SUBMIT Reward Bot



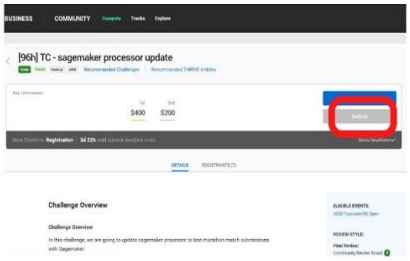
STEP: Browse

ACTION: Bot does not take direct action but is monitoring.



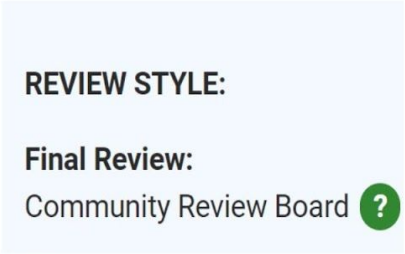
STEP: Register

ACTION: Bot records base point value for registering for a problem. Adds to users existing points and displays updated values. This step encourages developer participation.



STEP: Submit

ACTION: Bot records base point value for submission. This action does not take account the quality of the submission. However, it encourages participation through the solution process.

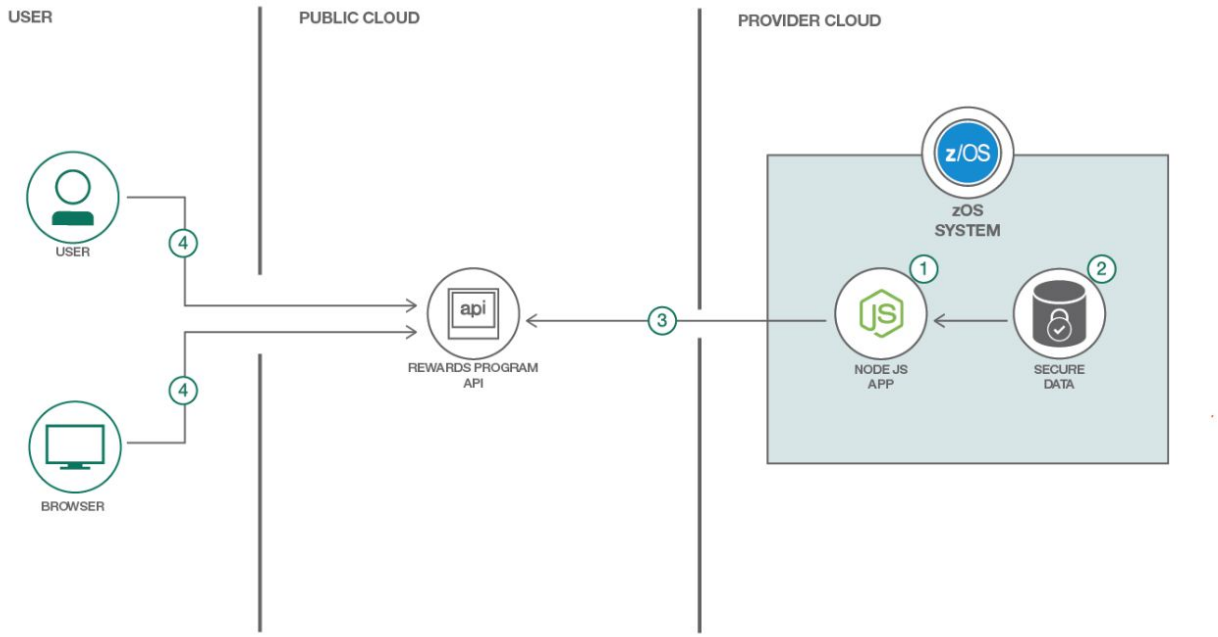


STEP: Review

ACTION: Bot uses review data from approval process to determine the quality of the submission. It awards points to the user based on scorecard. This encourages good quality of submission.

Architecture Design

Node.js Interaction with LoopBack Framework



(https://developer.ibm.com/technologies/node-js/patterns/create-rewards-program-apis/)

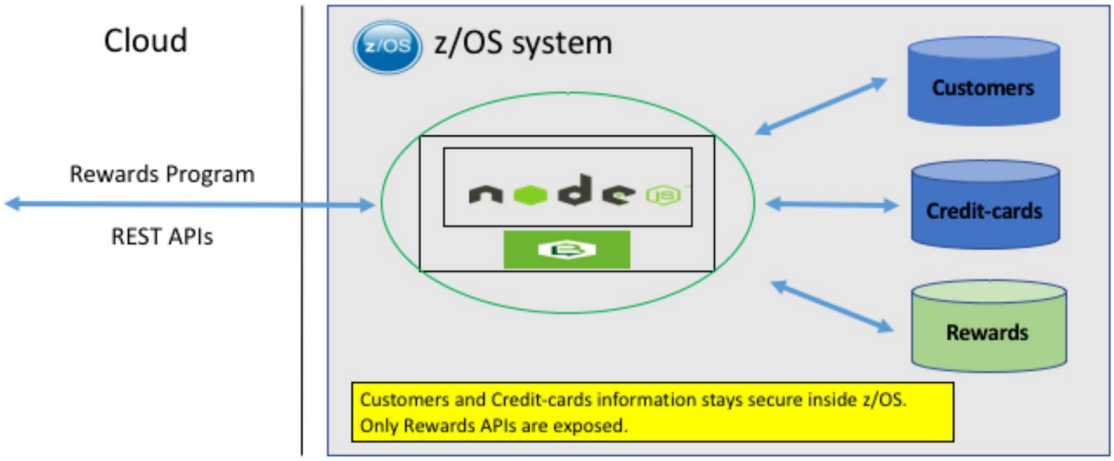
The bot will be created using a Node.js rewards program application using the LoopBack framework. Using a demonstration from IBM modeling a credit card rewards program, we will create the base workings for the bot, where the bot will monitor points required to be added, and be able to take action to award those points.

In IBM's scenario, "the TorCC credit card company holds data regarding the members, the credit cards and the reward programs. It provides a rewards program in which multiple members share the reward program. TorCC wants to expose APIs for frontend or mobile usage, to query and manage the reward program. In our example, we generate 4 APIs to deal with the rewards programs:

1. create a program
2. query the status
3. delete/close a program
4. claim points

These APIs cover the full spectrum of create, retrieve, update and delete (CRUD) functions."

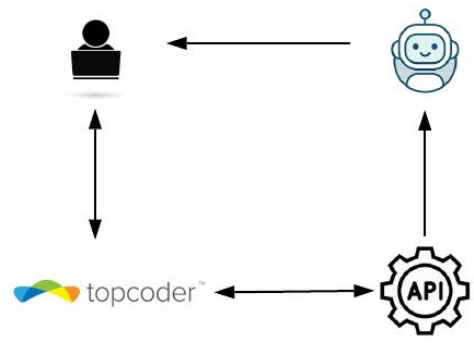
Node.js Interaction with Rewards Program APIs



(https://github.com/ibm/loopback-demo-zos)

Third party services that we may need to use include TopCoder's API. We want to use TopCoder's API to communicate community member's actions to our reward bot. The reward bot will then know which users to notify for reward points and what data to send to the database. Use of API integration and architecture within the project can be seen in the Node.js Interaction with Loopback Framework diagram. Additionally, interactions that the team would want to replicate can be seen in the diagram below.

API Architecture

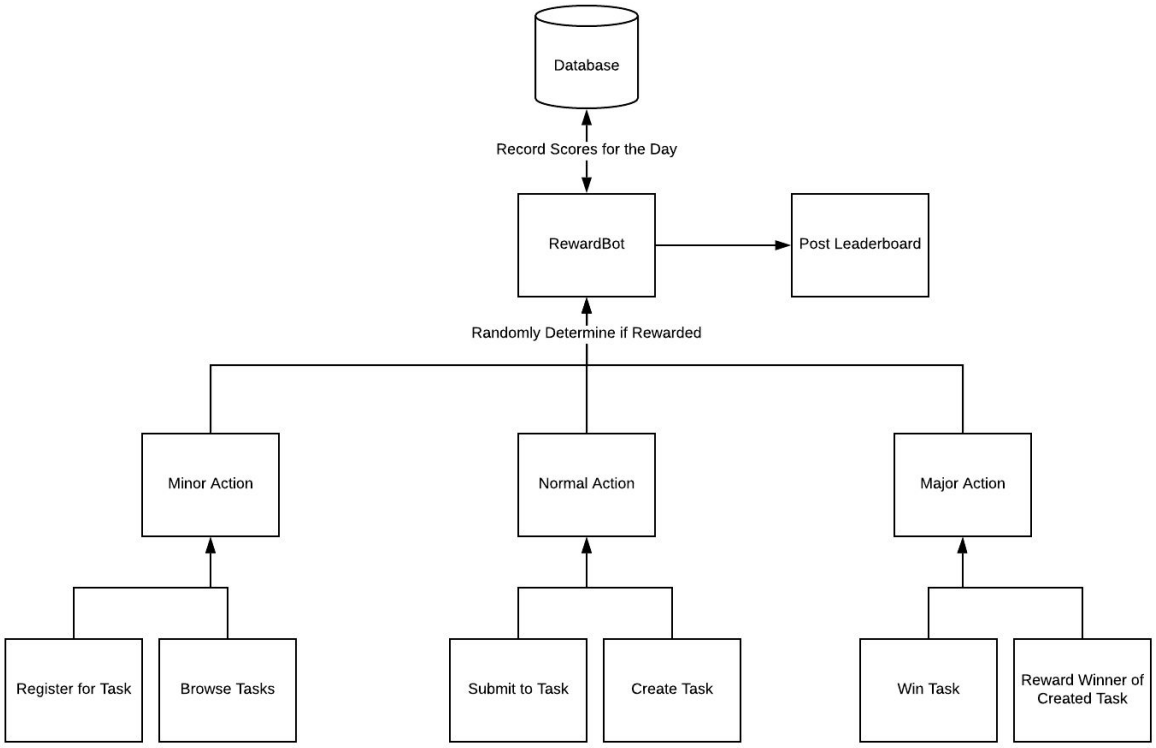


In the above diagram we see that the user and TopCoder website interact with each other. TopCoder also sends and receives information from the TopCoder API. The RewardBot will

receive information from the TopCoder API to be able to notify the correct users regarding reward points.

The RewardBot requires a database to store user information and their points earned for use in the leaderboards. Google Firebase can be used for this as a real-time database and back-end service. By using firebase the bot can easily update the database when users earn points and access the top scorers to display on the leaderboard.

General Architecture Overview: Actions



In order to ensure both the participation of the developer and the quality of the solution submitted, reward bot will monitor registering, completing, and the scorecard of tasks completed. The bot will store the point values and display the most current value to the user. It will need extremely limited database storage in our vision, since it will only require simple integer storage for each user.