SZCZEGÓŁOWY OPIS PRZEDMIOTU ZAMÓWIENIA

Załącznik nr 1 do Zapytania ofertowego nr 6/POPW.01.01.02-18-0013/17

Zamówienie realizowane w ramach Europejskiego Funduszu Rozwoju Regionalnego, osi priorytetowej I: Przedsiębiorcza Polska Wschodnia, działania 1.1 Platformy startowe dla nowych pomysłów, Poddziałania 1.1.2 Rozwój startupów w Polsce Wschodniej Programu Operacyjnego Polska Wschodnia 2014 -2020 (POPW).

Zapytanie ofertowe dotyczy **Rozbudowy technologicznej platformy Challengerocket składająca się z 4 części:**

1. Dodanie kontentu dotyczącego zadań programistycznych
2. Rozbudowa platformy o mechanizm auto-adaptacji poziomu trudności na podstawie dostarczonej specyfikacji

c) Dodanie zaawansowanego systemu raportowego dla rekrutera
d) Dodanie modułu oceny struktury kodu programistycznego w oparciu o metody sztucznej inteligencji

Zadanie może zostać wykonane w całości lub w poszczególnych obszarach/obszarze związanych z Państwa specjalizacją.

O startupie:

ChallengeRocket.com dynamicznie rozwijające się przedsięwziecie w obszarze HR-TECH i INNOWACJI o globalnym zasięgu. Zajmujemy się **kompleksowym i zautomatyzowanym podejściem do rekrutacji pracowników IT** (pozyskiwanie kandydatów i ewaluacja umiejętności) z wykorzystaniem metod sztucznej inteligencji. Świadczymy usługi doradczo - wykonawcze w zakresie **pozyskiwania innowacyjnych pomysłów** i **odkrywania talentów** z wykorzystaniem formuły hackathonów.

Gromadzimy społeczność twórców z całego świata oraz firmy i organizacje (rządowe i pozarządowe), które chcą do tych twórców dotrzeć ze swoimi wyzwaniami. Z ChallengeRocket współpracowały m.in. **Bank Zachodni WBK, Fortum czy NVIDIA.** Startup wyróżniony za innowacyjność 1-szą nagrodą na Europejskim Kongresie Gospodarczym w Katowicach (2017) oraz 1-szą nagrodą w ramach prestiżowego kongresu ABSL w Łodzi (2017).

Przykładowe hackathony, organizowane na naszej platformie przy naszym wsparciu:

- https://challengerocket.com/nvidia (hackathon globalny, w formule online, trwający obecnie na naszej platformie do lutego 2018)
- https://challengerocket.com/accorhotels (hackaton lokalny, w formule stacjonarnej)
- https://challengerocket.com/fortum (hackathon hybrydowy - 1-sza część online, 2-ga część stacjonarna)
- https://challengerocket.com/bankitup (hackaton lokalny, w formule stacjonarnej)

Publikacje na temat naszej platformy i osiągnięć: https://challengerocket.com/media.html

1. Ogólne wymagania

1.1. Wprowadzenie

Poniżej zostały opisane kluczowe wymogi oraz funkcjonalności systemu. Serwis challengeRocket umożliwia użytkownikowi start w wyzwaniach technologicznych (konkursy obsługiwane przez platformę). Klient firmowy powinien mieć możliwość tworzenia nowych wyzwań konkursowych lub wybór istniejących z biblioteki wyzwań i przekazywanie ich do swoich pracowników w ramach ćwiczeń lub używanie w ramach rekrutacji.
Zostanie dalej opisana perspektywa programisty (użytkownika który startuje w wyzwaniach) oraz firmy (która może konfigurować nowe wyzwania poprzez system CMS).

Językiem serwisu jest j.angielski. Serwis powinien być responsywny i większa część jego widoków powinna być dostosowana do widoków mobilnych.

W trakcie realizacji zlecenia Wykonawca może udzielać szczegółowych wytycznych w trakcie realizacji prac programistycznych oraz dostarczać szczegółowych widoków graficznych serwisu w postaci plików psd.

Zamawiający uzgodni z Wykonawcą poziomy kar umownych za niedopełnienie poszczególnych punktów umownych.

1.2. Reagowanie na błędy i dostępność serwisu

Wszystkie zauważony błędy systemu powinny być naprawione w przeciągu 48 godzin.
Błędy krytyczne systemy powinny być naprawione w przeciągu maksymalnie 24 godzin od czasu zgłoszenia.
Wykonawca gwarantuje dostępność serwisu na poziomie co najmniej 99,7% zarówno po stronie użytkownika ostatecznego oraz administratora.
Wykonawca nie odpowiada za błędy wynikające z dostępnością serwera, natomiast odpowiada za brak dostępności serwisu spowodowany wystąpieniem błędów uniemożliwiających jego prawidłowe funkcjonowanie lub ograniczoną dostępność.

1.3. Wymogi bezpieczeństwa

Wykonawca powinien gwarantować niezbędny poziom bezpieczeństwa systemu, dokonywać stałego monitoringu bezpieczeństwa przez cały okres rozwoju oraz 1 rok po jego wypuszczeniu.

W ramach realizacji zlecenia Wykonawca powinien wykonać zewnętrzne zlecenie audytu bezpieczeństwa z firmą audytująca wskazaną przez Zlecającego, a następnie naprawić wszystkie ewentualnie wykryte luki bezpieczeństwa wskazane w raporcie bezpieczeństwa.

1.4 Dostępność osób

Wykonawca powinien przedstawić Zlecającemu profile Project Managerów oraz programistów, którzy będą pracować nad zleceniem.

2. Perspektywa użytkownika systemu

Użytkownik może używać systemu jako pracownik firmy lub “osoba zewnętrzna”.

Użytkownik systemu może startować w szeregu wyzwań za pomocą których sprawdza swoją wiedzę oraz umiejętności programistyczne. Na podstawie realizacji danych zadań dostaje szczegółowy raport.

2.1 Zakładanie konta

Użytkownik może założyć konto używając email lub facebook.
Dostępna opcja przypominania hasła w przypadku logowania email.
Określone wymogi dot. bezpieczeństwa hasła (m.in. długość).
Konieczność potwierdzenia rejestracji przez email.

2.2 Uzupełnianie profilu

Użytkownik może uzupełniać swoje podstawowe dane dot. umiejętności, zainteresowań i celów jako programista.
Profil programisty jest widoczny pod stałym linkiem, który generuje się na podstawie nazwy użytkownika. Użytkownik może wprowadzać również swoje prace do profilu aby zaprezentować swoje portfolio (praca użytkownika to może składać się z galerii zdjęć, filmów oraz opisów).
Jeśli użytkownik wygra danych konkurs to informacja o tym fakcie będzie również widoczna w jego profilu.
Po zakończeniu wybranych konkursów użytkownik będzie mógł również pochwalić się tym i pokazać szczegółowe wyniki jakie udało mu się osiągnąć.

2.3 Koła “dev circles”

Użytkownik może zapisać się do danego “koła programistów” lub założyć nowe (zostaje wtedy jego administratorem).

Warunkiem założenia nowego koła jest dostępna wolna nazwa jego domeny w ramach serwisu.
Wyróżniamy koła publiczne i firmowe (jedna linia podziału).
Każde koło może być również otwarte na nowych członków (każdy może dołączyć) lub wymagać akceptacji nowego członka przez administratora koła.
Każde koło posiada swoją nazwę, opis i opcjonalnie lokalizacje. Użytkownik może wyszukiwać kół ”dev circles” znajdujących się w jego pobliżu. Administrator danego koła może tworzyć lub wybierać nowe wyzwania dla uczestników koła (sam mechanizm działania wyzwań opisany w innym punkcie dokumentu). Wewnątrz kół są również widoczne rankingi (osobno dla każdego konkursu) obejmujące tylko jego członków, których działanie zostanie opisane dalej.

Administrator koła może zapraszać do niego nowych członków podając ich adresy email.
Przykładowa treść emaila “username” has invited you to join his dev circle "Masters".
z opcjami do wyboru: Accept | Decline
Administrator koła powinien mieć widoczną listę osób, które zaprosić wraz ze statusem zaproszenia (Accepted / Rejected / Pending).
Administrator koła powinien móc zarządzać uprawnieniami innych osób w ramach koła oraz mieć możliwość aby wyłączyć danego uczestnika z grupy.

Wewnątrz kół użytkownicy mogą prowadzić dyskusje używając API slack (dla nowego poła powinien być automatycznie tworzony kanał gdzie zapraszani się na bieżąco wszyscy dołączający członkowie koła, członkowie powinni być automatycznie wykasowywanie kiedy opuszczają koło / zostają z niego wyrzuceni lub całe koło jest likwidowane)

Tworzenie nowych wpisów w ramach kanału i widoczność wszystkich wpisów powinna być możliwa zarówno z poziomu dowolnego klienta Slack jak i bezpośrednio z poziomu serwisu ChallengeRocket.

2.4 Przeglądanie wydarzeń konkursowych

Użytkownik może przeglądać wszystkie konkursy, sortować je lub stosować filtrowanie po wybranych kryteriach.

Dostępna jest także wyszukiwarka konkursów. Wyszukiwanie powinno działać w czasie wpisywania frazy w wyszukiwarkę. W przypadku podawania danej lokalizacji powinny być wyszukiwane również hackathony które dzieją się w pobliżu danego miejsca (np. użytkownik wpisuje “Gdańsk” do wyszukiwarki widzi w pierwszej kolejności konkursy mające miejsce w Gdańsku ale również powinien dalej zobaczyć te które mają miejsce w Gdynii).
Użytkownik powinien również w osobnej sekcji mieć widoczne wydarzenia, które dzieją się jego okolicy jeśli uzupełnił profil o swoją lokalizację oraz te które są spójne z jego zainteresowaniami jeśli je również uzupełnił w profilu.

Wyszukiwarka powinna działać niezależnie od tego czy nazwy miejscowości zostaną wpisane z lub bez użycia znaków diaktrycznych.

W ramach systemu powinien zostać zaimplementowany algorytm rekomendacyjny, które sugeruje użytkownikowi obszary konkursów jakie mogłyby go zainteresować na podstawie obserwacji

poprzednich partycypacji i zaangażowania w innych wyzwaniach. Algorytm powinien działać wg wytycznych Zlecającego.

2.5 Uczestnictwo w hackathonach stacjonarnych

2.5.1. Konfiguracja i widok danych konkursu

Użytkownik może zapisać się do takiego wydarzenia i wtedy trafia do oficjalnej listy uczestników widoczną dla organizatorów takiego hackathonu.
Każdy hackathon opublikowany na platformie posiada szereg sekcji opisanych dalej. Same wydarzenie jest publikowane w dedykowanej subdomenie i personalizowane brandingiem klienta. Widok powinien być zbudowany (i możliwy do konfiguracji poprzez system CMS)

Zlecający przekaże widok prezentujący wszystkie sekcje wchodzące w skład hackathonu stacjonarnego. W szczególności takie sekcje zawierają informacje na temat

- miejsce wydarzenia
- daty
- kryteriów aplikowania (kto może startować)
- pełnego harmonogramu (który powinien być generowany w formie tabelarycznej oraz ikonograficznej w ramach sekcji How to apply)
- jurorów (umożliwienie wgrywania nazwisk, zdjęć oraz krótkich opisów)
- kryteriów wyboru zwycięzców
- nagród (umożliwienie wprowadzenia listy i opisów szczegółowych nagród oraz opcjonalnych zdjęć widocznych w dymkach z szczegółowymi informacjami na temat poszczególnych elementów nagrody. Możliwe powinno być również określenie łącznej puli nagród)
- miejsca (wraz z zaznaczoną mapką, stylizowaną wg wytycznych Zlecającego)
- szczegółowego harmonogramu
- informacji na temat regulaminu wydarzenia (Terms & Conditions)

Oraz umożliwiać:

- opcjonalne prezentowanie galerii zdjęć z poprzedniej edycji konkursu
- updatowanie nowych informacji w sekcji updates
- prowadzenie dyskusji na dedykowanej części forum
- prezentowanie informacji o patronach wydarzenia (W tym patronach medialnych) poprzez danie możliwości wgrywania zdjęć i logotypów
- możliwość pobierania materiałów dotyczących wydarzenia (w tym materiałow dla mediów)
- ustawienie sekcji kontaktowej lub formularza kontaktowego (osobno dla każdego wydarzenia system śledzi i archiwizuje zapytania jego dotyczące).
- publikowanie informacji na temat wyników konkursu na osobnej dedykowanej konfigurowalnej podstronie (strona zawierałaby informacje na temat nagrodzonych prac, osób oraz komentarze jurorów na temat zgłoszenia, również konfigurowane z poziomu CMS)
- wyświetlanie listy i statystyk wszystkich uczestników konkursu

W przypadku konkursów kilkuetapowych system powinien wyświetlać złożona wersja harmonogramu wg wytycznych zlecającego.

System powinien automatycznie wykrywać na jakim etapie jest aktualnie konkurs i wyświetlać odpowiednią informacją statusową.

Wszystkie informacje zorganizowane są w osobnych sekcjach wg projektu graficznego i możliwe do customizacji przez system CMS. System powinien dawać firmie która publikuje danych konkurs możliwość ustawiania administratorów swojego profilu firmowego i wszystkich konkursów które są w jego ramach publikowane.

System na podstawie kompletu wprowadzonych danych powinien móc wygenerować stronę w formie podglądu (widoczną tylko dla osób znających linka) lub umożliwiać jej publikację.

Każda sekcja w ramach konkursu powinna być możliwa do włączenia / wyłączenia (ustawienia widoczności).

System powinien dawać również możliwość anulowania konkursu co powinno być zaznaczone odpowiednim komunikatem na stronie takiego konkursu.

2.5.2 Zapisy na hackathony stacjonarne

Na hackathony stacjonarne obowiązują zapisy. Uczestnik przy zapisie może być proszony o podanie szeregu informacji poprzez formularz.
Formularz jest konfigurowalny przez administratora konkursu, który ustawiać w nim może

- pola tekstowe
- pola wyboru
- pola uploadu dodatkowych plików

Czynność ta powinna być możliwa do realizacii w prosty sposób za pomocą kreatora.

Firma na bieżąco widzi jakie zgłoszenia od jakich użytkowników przychodzą i jakie dane są załączona i może akceptować lub odrzucać zgłoszenia.

Użytkownik dostaje powiadomienie odnośnie zapisania na hackathon oraz powiadomienie jeśli jego uczestnictwo zostanie potwierdzone (do decyzji klienta przez system CMS wg opisu powyżej).

2.5.3 Galeria prac i głosowania

Galeria prac - w ramach hackathonu powinno być możliwe przez użytkownika zgłoszenie swojej pracy do galerii. Praca uzyskuje swoją osobą podstronę. Użytkownik dostaje email potwierdzający zarówno wysłanie jego pracy jak i status moderacji pracy. Przy okazji zgłoszenia pracy użytkownik może być poproszony o wypełnienie formularza który jest konfigurowalny przez administratorów konkursu w sposób opisany w poprzednim punkcie.

Prace mogą być ukryte (widoczne tylko dla administratorów konkursu w trakcie oceny) jak i upublicznione w galerii.

Administratorzy konkursu powinni mieć możliwość moderowania prac.
Pod opublikowanymi pracami powinna być możliwość komentowania.

Uczestnicy powinni również móc głosować na pracę w wyznaczonych terminach jeśli taka opcja zostanie włączona przez system CMS. Głosowanie może być możliwe bezpośredni przez serwis lub przez system facebook. W głosowaniu bezpośrednim mogą brać udział wszyscy użytkownicy lub tylko Ci wchodzący w skład danej firmy jeśli tak zostanie skonfigurowane w panelu w przypadku konkursu firmowego.
Wykonawca powinien zaproponować i wdrożyć podstawowe rozwiązania mające na celu identyfikacje botów oddających głosy w sposób automatyczny.

2.6 Uczestnictwo w hackathonach kodu

Szczególnych rodzajem wyzwań są konkursu gdzie w odpowiedzi na wyzwanie wpisuje się kod bezpośrednio w edytor programistyczny w ramach platformy ChallengeRocket.

Hackatonu kodu powinny być publiczne lub dostepny tylko z panelu firm, dla programistów wchodzących w skład ich kół firmowych. W ramach konfiguracji hackatonu kodu administrator powinien móc opisywać wyzwanie oraz dane na których powinny być przetestowane programy pisane przez użytkowników (zestawy danych wejściowych jakie programy będą po kolei wczytywać oraz prawidłowe odpowiedzi jakie powinny być zwracane dla tych danych lub wzorzec prawidłowych wypowiedzi zapisany w formie wyrażeń regularnych).

Wykonawca powinien umożliwić z poziomu serwisu kompilowanie kodu przynajmniej 12 języków programowania wskazanych przez Zlecającego (m.in. C, C++, Python 2, Python 3, Java, Go, Basic) oraz interpretowanie języka javascript.

Kod powinien być wykonywane w pełnej izolacji (wirtualizacja) i automatycznie zakańczany po osiągnięciu limitu czasu wyznaczonego na dane zadanie. System powinien móc liczyć czas wykonania kodu oraz czas pracy nad nim.

Powinna być dokonywana weryfikacja prawidłowości działania kodu pod kątem zwracania prawidłowych wyników na zadane zbiory testowe w zadanym czasie (zbiory powinny być wczytywane przez standardowe wejście z wyjątkiem języka javascript).

Jeśli test jest wykonywane wewnątrz firmy przez rekrutera to powinien on dostać pełny raport z jego realizacji w formacie widocznym na ekranie oraz w formie raportu pdf.

System w ramach realizacji hackatonów kodu powinien również umożliwiać

- dzielenie się rozwiązaniami kodu pomiędzy uczestnikami danego koła
- publikowanie wzorcowych rozwiązań

- przyznawania punktów w zależności od ilości testów które zaliczył dany kod (system punktowy działający na bazie szczegółowego modelu przekazanego przez Zlecającego)
- przyznawania punktów z pierwsze miejsca (najlepsze rozwiazania w ramach konkursu firmowego)
- przyznawanie dodatkowych punktów na podstawie indywidualnej oceny administratora koła.
- ustalanie limitu czasu wykonania pojedynczego kodu (automatyczne zakończenie po przekroczeniu) oraz maksymalnej zajętości pamięciowej danego rozwiązania.

Każdy konkurs powinien mieć sekcję Leaderboard (użytkownik może uczestniczyć w danym wyzwaniu zarówno w jako w wyzwaniu otwartym jak i w ramach danego koła, wtedy widzi rankingi zarówno otwarte jak tylko ograniczone do członków danego koła).

2.7 Uczestnictw w pojedynkach kodu (gry botów)

Pojedynki kodu są szczególnym rodzajem hackatonów kodu, w których system powinien umożliwiać uruchamianie napisanych przez użytkownika kodów które wykonują ruchy w grze strategicznej (np. szachy) i wymianę między nimi danych na temat ruchów umożliwiających odbycie pełnej rozgrywki.

W ramach testu tego modułu użytkownik powinien zaprogramować system oraz boty umożliwiające grę w :

- szachy (przykładowy bot na podstawie przeszukiwania wgłąb drzewa gry z ucinaniem gałęzi o parametryzowalnej głebokości przeszukiwania)
- warcaby

Użytkownik powinien móc uruchomić grę w 2 trybach

- szybkim (pokazującym odrazu wynik rozgrywki)
- wolnym (gdzie poszczególne ruchy są wizualizowane)

W ramach danej gry powinien zostać tworzony bardziej rozbudowny leaderboard pokazujący nie tylko finalne wyniki ale także historię rozgrywek pomiędzy użytkownikami.

Użytkownik może wyzywać innego użytkownika do gry lub powinien być uruchomiony system umożliwiający

- odbycie rozgrywki w trybie ”każdy z każdym” w przypadku mniejszej ilości graczy (do 5 w danej grupie)
- obycie rozgrywki w systemie turniejowym

Na tej bazie budowany jest finalny ranking. Użytkownik który wykona niedozwolony ruch jest dyskwalifikowany z danej rozgrywki.

Użytkownik w ramach testu swojego kodu powinien móc także zagrać z botem obsługiwanym przez komputer. W grze powinien obowiązywać również limit czasu na pojedynczy ruch. Po jego przekroczeniu użytkownik jest dyskwalifikowany i automatycznie przegrywa daną grę.

2.8 Zadania testowe

Oprócz wykonywania zadań zw. z pisaniem kodu system powinien umożliwiać obsługę zadań quizowych.
Zadania takie są konfigurowane przez system CMS i następnie publikowane w formie otwartej (każdy zarejestrowany programista może przystąpić) lub być dostępne na zaproszenia.

Powinny być obsługiwane:

- pytania jednokrotnego wyboru
- pytania otwarte (gdzie ocena odbywa się w sposób manualny przez administratora konkursu)

Do każdego pytania system powinien:

- umożliwiać dodanie poziomu trudności oraz punktów za prawidłowe rozwiązanie
- umożliwienie określenia kategorii pytania
- towarzyszących multimediów (zdjęcia, wstawki kodu, filmiki)

Powinien umożliwiać obsługę limitu czasowego (konfigurowalny)

2.9. Adaptacja poziomu trudności

System powinien umożliwiać realizację quizów z samo-adaptującym się poziomem trudności wg szczegółowych wytycznych zamawiającego.
System na bieżąco powinien obserwować odpowiedzi jakich udziela użytkownik na różne rodzaje pytań / jak rozwiązuje różne zadania. Na podstawie tego jak radzi sobie z rozwiązaniami powinien zmieniać poziom trudności.

2.10 Uwagi końcowe

System powinien obsługiwać zdarzania powrotu do wypełnianego quizu/ zadania kodu przerwanego w sposób nieoczekiwany (problemy z połączeniem sieciowym,

System powinien informować o problemach z połączeniem sieciowym w trakcie realizacji zadań przez użytkowników.

W przypadku zaproszeń do wykonania testu możliwy jest dodatkowy parametr precyzujący kiedy test będzie ważny (po danym okresie nie można do niego wejść).

2.11 System raportowy dla rekrutera

Po realizacji testów/hackatonów administrator powinien widzieć kompletne wyniki i raport z wykonania pokazujący obszary nad którymi spędzono najwięcej czasu i które sprawiły największą trudność.

System powinien umożliwiać zaawansowane zdolności raportowania dot. wyniku testu pojedynczego użytkownika umożliwiające m.in. stwierdzenie

- jaki był czas wykonania zadania (bezpośrednia kalkulacja czas zakończenia - czas uruchomienia)
- stwierdzenie jak wygląda wynik użytkownika na tle innych wyników (o ile % wynik użytkownika jest wyższy od średniej serwisu / o ile % wynik użytkownika jest lepszy od średniej w jego firmie)
- opcjonalne statystyki: jak wygląda wynik kandydata w porównaniu do innych osób z tym samym doświadczeniem.
- pokazanie jaki jest wynik użytkownika w różnych obszarach (poszczególne pytania mają przypisane kategorie. Rekruter widzi jakie jest jego wynik w poszczególnych kategoriach w ramach danego testu).
- pokazanie czasu wykonania zadania vs czasu zadeklarowanego użytkownika na rozwiązanie danego zadania
- pokazanie ocen automatycznych razem z ocenami manualnymi
- umożliwienie przeglądu pytań i udzielonych odpowiedzi
- dla każdego pytania umożliwienie sprawdzenia ile % osób prawidłowo odpowiedziało na zadane pytanie (i porównanie do użytkownika którego wynik widzimy).

Podstawowa wizualizacja wspomnianych wyżej statystyk w formie wykresów.

Dodatkowo powinna być możliwość przedstawiania raportów łącznych pokazujących:

- historię wyników pojedynczego użytkownika (w formie tabelarycznej / wykres)
- podsumowanie wyników danego użytkownika w różnych obszarach
- porównanie kilku użytkowników biorących udział w projekcie (w formie tabelarycznej / wykres). Porównanie łącznego wyniku oraz historię odpowiedzi dla każdego użytkownika na każde pytanie quizowe w formie tabularycznej.
- wskazanie jaka jest najlepsza osoba w danej grupie (ranking)

Dla określonych konkursów realizowanych w ramach firmy wskazanie jaki był zwycięzca danego wydarzenia / jaki średni wynik itp.

Sam programista, który przeprowadzał test, widzi część wyżej wspomnianych statystyk.

2.12 Ocena struktury kodu programistycznego w oparciu o metody sztucznej inteligencji

Wykonawca powinien zaproponować rozwiązania umożliwiające podstawą ocenę struktury kodu i identyfikację podstawowych błędów w kodzie wskazujących na złe nawyki programistyczne.
W szczególności wykonawca w realizacji tego działania może zintegrować istniejące narzędzia typu lintern.
Rozwiązanie powinno działać przynajmniej dla 4 jezyków programowania.

3. Zaproszenia do realizacji testów.

3.1. Uczestnictwo w teście przez programistę

Użytkownik może uczestniczyć w hackathonie testowych

- po otrzymaniu zaproszenia od rekrutera do testu zamkniętego (zaproszenie email)
- po otrzymaniu zaproszenia przez administratora dev circle
- po zapisaniu się do testu otwartego

4. Uzyskanie certyfikatu przez programistę

4.1 Przyznawanie

Niektóre testy lub wyzwania powinny być skończone otrzymaniem specjalnego certyfikatu przez programistę. (opcja konfigurowalna z panelu CMS)

4.2. Profil programisty

Certyfikaty z szczegółowymi wynikami powinny być widoczne z poziomu profilu użytkownika jeśli wyrazi on na to zgodę

5. Doświadczenie klienta, zamówienie i zarządzanie kontem

5.1 Zamówienie abonamentu

Zamówienie jest realizowane przez wybór któregoś z 3 wersji abonamentu lub wybór 2 tygodniowego okresu testowego (Try for free). Edycja parametrów powinna być możliwa przez system CMS administratora serwisu.

Dla klienta niezależnie od wybranej opcji abonamentu płatnego doliczane jest zawsze 2 tygodnie (14 dni) okres bezpłatnego.

Po wyborze jednego z 3 typów abonamentów (które będą się różnić dostępnymi opcjami opisanymi dalej) zostajemy przekierowani do strony płatności, gdzie możemy wprowadzić dane karty kredytowej.

Możliwe jest również wpisanie promo code (kodu promocyjnego), które ogranicza cenę abonamentu o dany procent. Jeśli cena zostanie obniżona do zera to nie pokazujemy wtedy

(kody zniżkowe działają dla wszystkich abonamentów i umożliwiają dostęp przez 1 miesiąc do serwisu).

Dla administratora systemu powinna być możliwe wygenerować losowe kody (np. administrator chce 10 kodów na dany % zniżki). Istnieje czas: 30 dni na wykorzystanie danego kodu. Po tym czasie kod jest nieaktywny.

5.2. Przedłużenie abonamentu

Po roku czasu abonament musi zostać przez klienta przedłużony. Klient dostaje wcześniej odpowiednie powiadomienia generowane przez system.

5.3. Widok trwających abonamentów

Administrator systemu powinien mieć widoczną listę osób które kupiły abonamenty lub sam trial.

W tabelce powinny być dostępne następujące informacje:

- firma (nazwa) + dane osoby kontaktowej
- email
- rodzaj wykupionego abonamentu i wartość
- do kiedy czas trwania abonamentu lub okresu testowego
- czas rozpoczęcia (kiedy pierwszy raz klient aktywował konto)
- łączna kwota jaką dany klient zapłacił od początku swojej aktywności (wartość klient)

Po kliknięciu na przycisk Details powinniśmy otrzymać całą historię aktywności tego klienta oraz historię jego zakupów.

5.4. Generowanie faktury

System powinien umożliwiać wygenerowanie faktury klientom którzy zapłacili za abonament. (klient podaje dane do faktury). Z poziomu panelu klient powinien mieć dostęp do historii wszystkich swoich operacji oraz listę wszystkich faktur. Z tego poziomu powinien móc również zgłaszać problemy z płatnościami.

5.5 Zarządzanie kontem i przedłużanie abonamentu

5.5.1 Zarządzanie dostępem (right management)

Użytkownik konta firmowego powinien mieć możliwość ustawiania administratorów systemu którzy mają dostęp do tworzenia i zarządzania kołami programistów (dev circles). Mogą oni dodawać (poprzez wysyłanie zaproszeń) i kasować z nich użytkowników oraz ustalanie nowych zadań.

5.5.2. Pobieranie płatności i przedłużanie abonamentu

Abonament jest zamawiany na 12 miesięcy. Płatność powinna być automatycznie pobierana przez każdy kolejny miesiąc na jego początku.

Po roku czasu użytkownik powinien ręcznie przedłużyć abonament (automatycznie są wysyłane przypomnienia o płatności).

W przypadku problemów z pobieraniem płatności użytkownik powinien dostawać automatyczne powiadomienia i powinien mieć możliwość aby ręcznie zmodyfikować dane karty. Jeśli tego nie zrobi po 14 dniach blokowany jest dostęp do jego konta (wciąż może modyfikować dane i odzyskać swoje dane i dostęp do konta po pobraniu płatności).

5.5.3. Promocje - oferty

Poprzez system administracyjny CMS powinniśmy mieć możliwość ustawiania promocji na abonamenty.

5.5.4. Przypomnienia

Wszystkie przypomnienia będą generowane drogą emailową oraz opcjonalnie SMS (do decyzji szczegółowej Zamawiającego)

6. Kontent dot zadań programistycznych

Wykonawca powinien w ramach realizacji zlecenie stworzyć kontent umożliwiający testowanie umiejętności programistów w postaci

- pytań “quizowych”
- wyzwań programistycznych

Wszystkie zadania oraz pytania quizowej powinny być napisane w języku angielskim.

6.1. Pytania quizowe

Pytania powinny być relewantne w takim sensie, iż możliwie w największym stopniu na podstawie odpowiedzi na nie można stwierdzić jakie są realne umiejętności programistyczne danej osoby (w szczególności pytania nie powinny odnosić się wprost do elementów których programiści nie pamiętają (np. nazwy rzadko używanych funkcji) a zwyczajowo sprawdzają w dokumentacji.

Do każdego pytania powinna być przypisana kategoria oraz jego poziom trudności. Pytania powinny jeśli to możliwe pokrywać maksymalny obszar danego języka programowania lub technologii w zakresie w jakim jest ona praktycznie używana

Każde pytanie powinno być uzasadnione merytorycznie i na szczególną prośbę Zamawiającego podane być uzasadnienie zastosowania tego pytania, wytłumaczenie jaką rzecz sprawdza oraz uzasadnienie poziomu trudności.

6.2. Wyzwania programistyczne/wyzwania kodu

Wykonawca powinien ułożyć autorskie zadania kodu na wzór wyzwań używanych przez Olimpiadach Informatycznych dla Studentów (https://pl.wikipedia.org/wiki/Mi%C4%99dzynarodowa_olimpiada_informatyczna, http://oi.edu.pl/l/linki_olimpiady/) o różnych poziomach trudności. W odróżnieniu od zadań typowych dla olimpiad informatycznych wyzwania powinny w większym stopniu kłaść nacisk na weryfikację inżynieryjnych umiejętności programistów.

Każde zadania powinno określać problem do rozwiąania, podawać serię danych wejściowych na których dane zadanie jest testowane oraz określać pożądane dane wyjściowe jakie powinien zwracać prawidłowo napisany kod rozwiązania zadania. Dane powinny pokrywać przypadki brzegowe problemu i być na tyle obszerne aby umożliwić zbadanie złożoności obliczeniowej kodu.

Wykonawca powinien określić i rozpisać wzorcowe rozwiązanie w jednym lub kilku językach programowania obsługiwanych przez platformę (do szczegółowego uzgodnienia ze Zlecającym).

Ilość quizów wiedzowych: min 40 (każde minimum 7 pytań), łącznie minimum 400 pytań.

Ilość zadań programistycznych do rozwiązania: min 25

7. Analityka i optymalizacja struktury serwisu

Wykonawca powinien dokonywać modyfikacji wyglądu serwisu przez okres 1 roku wg wytycznych zlecającego pod kątem optymalizacji konwersji na podstawie uwag przekazywanych przez Zlecającego. Zmiany będą dotyczyły głównie rozmieszczenia poszczególnych elementów.

System powinien być zintegrowany z systemem Google Analytics wg wymogów klienta.
System powinien być zintegrowany z systemem SalesManago wg. szczegółowych wymogów klienta.