



Micro-Task and Earning Platform

assignment12_category_0019

Hello Candidate,

Welcome to the job assessment for the position of Junior MERN Stack Developer at **PicoWorkers**. We are excited to evaluate your skills and capabilities in developing robust, d efficient web applications using the MERN stack. This assessment is designed to gauge your proficiency in MongoDB, Express.js, React.js, and Node.js, as well as your problem-solving abilities, coding practices, and overall technical acumen.

This assessment consists of a project that you will need to complete using the MERN stack. This project is designed to evaluate your MERN-Stack technologies and skills.

Key Rules:

- Include a minimum of 20 notable GitHub commits on the client side
- Include a minimum of 12 notable GitHub commits on the server side
- Add a meaningful readme.md file with the name of your website, Admin username, password and live site URL. Include a minimum of 10 bullet points to feature your website.
- Make it responsive for all devices. You need to make it responsive for mobile, tablet and desktop views. Make the dashboard responsive as well.
- After reloading the page of a private route, the user should not be redirected to the login page.
- Use the Environment variable to hide the Firebase config keys and Mongoddb credentials.
- Don't use any Lorem ipsum text in your website.

Tips for You

- Read and understand the assessment theme carefully.

- Select a design and stay stuck with the same color and layout.
- Don't go through the whole requirement at once. Complete requirements one by One.

Good Luck! We wish you the best of luck and look forward to seeing your technical prowess and innovative solutions

Assessment Overview

The Micro Tasking and Earning Platform is designed to provide users with opportunities to complete small tasks and earn rewards. The platform accommodates 3 distinct roles : Worker, Task-Creator, and Admin. Each role is tailored with specific functionalities to ensure seamless task management, task creation, and platform administration.

1. **Workers:** Workers are users who engage with tasks posted on the platform to earn rewards. Capabilities are
 - Can View all available tasks along with detailed task descriptions and requirements.
 - Can Submit completed tasks for review and earn coins upon approval by Task-Creators.
 - Can withdraw their coin.
 - Receive notifications regarding earnings and withdrawals.
2. **Task-Creator:** Task-Creators are users responsible for creating tasks, reviewing task submissions, and managing task-related transactions. Capabilities are
 - Can Create tasks with specific instructions, deadlines, and reward amounts.
 - Can Review task submissions from Workers and approve/reject them accordingly.
 - Can Pay Workers for completed tasks automatically using platform coins.
 - Can Purchase coins from the system to facilitate task payments.
 - Can Report Workers to Admin for any violations or issues.

3. **Admin:** Admin oversees the overall functioning of the platform and ensures smooth operations. Capabilities are

- Modify user roles as necessary, including granting Task-Creator status and addressing reported issues.
- Provide feedback on Task-Creator reports and take appropriate actions.
- Manage platform integrity by deleting tasks, users, or any other system elements as needed.

Similar applications also exist on the market like picoworkers, microworkers, crowd-space, etc. you can search and get ideas from them also. Here is a technical description about the Data Modeling for this system. [📄 Database Design || Micro-Task and Earning Platform](#)

Main Requirements

1. Layout

The layout for the Micro Tasking and Earning Platform will be divided into two primary structures: **Basic Layout** and **Dashboard Layout**. We will talk about Dashboard layout after 4th Requirement

Basic Layout
Navbar
Sections Based on Routes
Footer

*** Design an Error Page and a Forbidden Page for Authorization purposes and invalid routes.

2. Navbar and Footer

Navbar will serve as the primary navigation tool for users, offering easy access to the platform's main features and functionalities. Navbar will contain following Navigations

For Not Logged in Users

- Website Name / Logo (By clicking, it will redirect user to home page)
- Login
- Register
- Watch Demo (This Button will redirect you to a youtube Video)

For Logged in Users

- Website Name / Logo (By clicking, it will redirect user to home page)
- Dashboard
- Available Coin + User Profile and Logout Button

Footer will contain website Logo , clickable social media icons which will redirect users to your profile such as linkedin , facebook , github , etc.

3. Home Page

The Homepage will be the first impression of the platform, so it needs to be engaging, informative, and visually appealing.

- **Hero Section**

It will contain a slider with three banners. Use react-responsive-carousel / Swiper Slider. Each slide will contain a different heading and Title. You can also use a background video instead of a slider.

- **Features Section:**

Highlights the key features of the platform. use icons / pictures and descriptions for each feature, such as "Earn Coins by Completing Tasks," "Create and Manage Tasks," and "Secure Payments."

- **How It Works Section**

Includes three main steps with icons and descriptions, such as "Register," "Complete Tasks," and "Earn Rewards."

- **Top Earners**

Show Top 6 Workers data who have earned maximum coins. Show their picture , available coins , task completion from database

- **Testimonial Section:**

Displays feedback of satisfied users in a slider format. Includes user photos, names, and brief quotes about their positive experiences. This section will be static. Use a swiper slider to build this section.

4. User Authentication System

The user authentication system will manage the registration and login processes for the platform. It ensures that users can securely access the platform, and the appropriate roles and permissions are assigned. Here's a detailed description of the system:

1. Registration Page

Here, Users can create an account by providing necessary information. After registration, the worker will get 10 coins and taskCreator will get 50 coins by default.

There will be 2 Types of Registration methods that have to be implemented here.

A) Form with Input Fields:

- Name
- Email
- Profile Picture URL (implement imageBB for uploading if want to get challenge mark)
- Password
- Drop-down for select the role
 - Worker
 - TaskCreator

*** Implement Input validation for email format and password strength. show Error messages for invalid input, such as existing email .

B) Google Login:

Implement Google Login. The Logged in users by google will be "worker" by default.

After Successful Registration you need to store user information and coin into the database.

- If the user is a Worker. Then coin : 10 .
- If the user is a TaskCreator. Then coin: 50.

if your collection already has the user Info Don't insert the user-info again. Add validation in your server.

2. Login Page

Here, Users can Sign in by providing necessary information. There will be 2 Types of Login methods that have to be implemented here.

- Users can log in using their registered email and password.
- Google Sign-In option for quick authentication.


*** Implement Input validation for incorrect email password. After successful Login, redirect the user to the Dashboard.

*** After Login / Registration you have to implement JWT on the server and save an access-token for users in his browser local-storage.

DASHBOARD

Layout

Dashboard will be Look Like This below table

Logo	Available coin userImage userRole userName	 Notification
Navigation	Sections Based on Routes	
	Footer	

Dashboard will Show Following Navigation based on user Role.

Worker	TaskCreator	Admin
Home	Home	Home
TaskList	Add new Tasks	Manage Users
My Submissions	My Task's	Manage Task
withdrawals	Purchase Coin	
	Payment history	

Dashboard For Worker

1. Worker Home

States

Worker will see the his available coin, Total Submission (Count of all submission made by worker) , Total Earning (sum of payable_amoun of the worker where status is approved)

Approved Submission

Worker will see all the submission made by him where the status is “approved ” in a table format from submission collection with following information

- task_title
- payable_amount
- creator_name
- status

2. TaskList

In This Route, Worker will see All the tasks where the task_quantity is greater than 0 with Following information

- task_title
- creator_name

- completion_date
- payable_amount
- task_quantity
- View Details Button

Data will be in card format. By clicking view Details navigate workers to the task details route.

3. Task Details

show all the information of the Task and a submission form in this Route.

The Submission form will contain 1 input field(text-area) name submission_Details. After submitting the form insert following informations in the submission Collection

- task_id
- task_title
- task_detail
- task_img_url
- payable_amount
- worker_email
- submission_details
- worker_name
- creator_name
- creator_email
- current_date
- status (pending)

4. My Submission

Show all the submissions information from submissionCollection where the workerEmail matched with the current worker Email . show data in a tabular form.

5. Withdrawals

20 Coins = 1 Dollar.

This Route will show how much money a worker can withdraw. Suppose if a worker has 300 coins then he can withdraw a maximum 15 dollars.

Maximum Withdrawal Amount

Show user his Maximum WithDraw Amount (dollar)

Withdrawal Form

- Coin To WithDraw (Number)
- withdraw_amount (Number) (Not editable . It will change when the coin to withdraw field changes. 20 coin = 1 dollar)
- Select Payment System (DropDown) (Bkash, Rocket, Nagad)
- Account Number
- WithDraw Button

if The Amount is greater than maximum withDraw account then reject his request and show him an alert.

else, insert all the information in the withdrawCollection with following info

- worker_email
- worker_name
- withdraw_coin
- withdraw_amount
- payment_system
- withdraw_time

Dashboard for Task-Creator

1. Task-Creator-Home

States

Task-Creator will see the his available coin, pending Task(sum of all task_quantity of his added Tasks), total payment paid by user

Task To Review

Task-Creator will see all Review requests of his tasks where the status is "pending" in a table format from submission collection with following information

- worker_name and worker_email
- task_title
- payable_amount
- View Submission Button(will open a modal and show the submission detail)
- Actionable Buttons
 - Approve Button
 - Reject Button

*** On Clicking Approve Button increase payable amount coin for the workers and change the SubmissionStatus to "approve" to the submission collection

*** On Clicking Reject Button change the status to "rejected" to the submission collection

2. Add new Tasks

This section will contain a Form with following input fields

- task_title
- task_detail
- task_quantity (number)
- payable_amount (per Task) (number)
- completion_date
- submission_info
- task_image_url (implement imageBB for uploading if want to get challenge mark)
- Add Task

On Clicking Add Task

***check if (task_quantity* payable_amount) is greater than users available count then through an alert "Not available Coin. Purchase Coin " .

Else, it will add the following input field information and creator_email , creator_name, current_time to the Task Collection of system Database and reduce (task_quantity* payable_amount) from users' available coins

3. My Task's

In this section the user will show all the tasks he added in descending order based on Time in a table format.

- Show The task_title , task_quantity, payable_amount, update and delete button
- onClicking update users can update the Title , TaskDetail and submission Count.
- onClicking Delete, delete the task from task Collection. And Increase the (task_quantity* payable_amount) coin in his available coin

4. Purchase Coin

From This Route User can purchase coins. Implement a stripe based payment system on this route.

Payment info

Show users 4 card with following info

- 10 coins = 1 dollar.
- 100 coins = 9 dollars.
- 500 coin = 19 dollars
- 1000 coin = 39 dollar

On Clicking a specific card redirects the user to pay a specific amount.

After successful payment , add the payment info into paymentCollection and increase TaskCreator Coin.

5. Payment History

Show All the payment made by the taskCreator in the payment history route in a tabular format.

Dashboard for Admin

1. Admin-Home

States

admin will see the count of total users , total coin, total payments

Withdraw request

Admin will see all withdrawal requests from withdrawCollection made by users in a table format with following information

- worker_name
- withdraw_coin
- Withdraw amount
- Payment Number
- Payment_system
- withdraw_time
- Payment Success Button

After clicking the payment success button ,data will be deleted from the withdrawal collection. And the user coin will be deducted by withdraw_coin from withdrawal collection

2. Manage Users

The section will show a table of all users who have the role “worker” with display_name, user_email, photo_url, role , coin and some actionable button

- remove (will delete user from the database)
 - By clicking Remove user will be deleted from the server.
- Update Role (Dropdown field. On change it will change the role of user)
 - Admin

- Task-Creator
- Worker

3. Manage Tasks

User will see The TaskList in a table format with following information

- task_title
 - TaskCreator Name
 - Task Quantity
 - Coin_Needed
 - Availability
 - View Task Icon (on Clicking It will open Modal and show Tasks details)
 - Delete Task (By clicking Task will be deleted from database)
-

Challenges

1. Secure Authorization

Implement Role Based Authorization for users. Create middleware for Worker, Admin and TaskCreator.

- For No Token send 401 status to the client. And logout the user immediately.
- For invalid Token send 400 status to the client. And logout the user immediately.
- For Role Based Authorization send 403 Status to the client and redirect the user to the forbidden route.

2. Notification System

Implement a notification System for Any operation on the Submission Collection and userCollection . Suppose if a taskCreator changes a particular task status on submission collection then add a notification object in the notification collection.

Example

```
{  
  message: " you have earned {payable_amount} from {taskCreatorName} for  
  completing {taskTitke}  
  ToEmail: {workerEmail}  
  Time: new Date()  
}
```

On Clicking the notification icon , show all the notifications where the email of the "toEmail" key matched with the current user email sorted in descending order.

Show notification in a floating pop up. By clicking anywhere of the page pop up will be hidden

3. Add Pagination on My Submission Collection for Worker

Implement pagination on the My Submission Route.

4. Image Uploading System with imageBB

Implement image uploading System with imageBB on registration and Add new Task Route.

5. Implement React Count Down on Task Detail Route

Implement This Package <https://www.npmjs.com/package/react-countdown> on Task Detail Route Based on the completionDate . take help from chatGPT / see the Demo.

Additional information:

- You can host images anywhere.
- You can use vanilla CSS or any library.
- Try to host your site on Firebase (Netlify hosting will need some extra configurations)
- Make Sure you deploy server-side and client-side on the first day. If you have
- any issues with hosting or GitHub push, please join the "Github and deploy" related support session.

Some Guidelines:

1. Do not waste much time on the website idea. Just spend 15-20 minutes deciding, find a sample website, and start working on it.
2. Do not waste much time finding the right image. You can always start with a simple idea. Make the website and then add different images.
3. Don't look at the overall task list. Just take one task at a time and do it. Once it's done, pick the next task. If you get stuck on a particular task, move on to the next Task.
4. Stay calm, think before coding, and work sequentially. You will make it.
5. Be strategic about the electricity issue.
6. use chat gpt to generate JSON data. You can use chatGPT for Other purposes as Well.

What to submit:

Assignment Category: assignment12_category_0019

Admin email:

Admin password:

Front-end Live Site Link:

Client Side Github Repository Link:

Server Side Github Repository Link:

Optional

You can also Add following features if you implemented it already

1. **Automated Email Notifications:**
 - Set up automated email notifications for various actions (e.g., task approval/rejection, payment confirmation, withdrawal processing).
 - Use services like SendGrid or AWS SES for sending emails.

2. Advanced Search and Filter Functionality:

- Implement a comprehensive search and filter system for tasks. Users should be able to filter tasks based on criteria like task type, deadline, reward amount, and status.
- Use MongoDB's aggregation framework for efficient querying and filtering on the server side.

3. Report System for invalid Submission

- Implement a reporting system for invalid submission. So that admin can take proper action on the user.