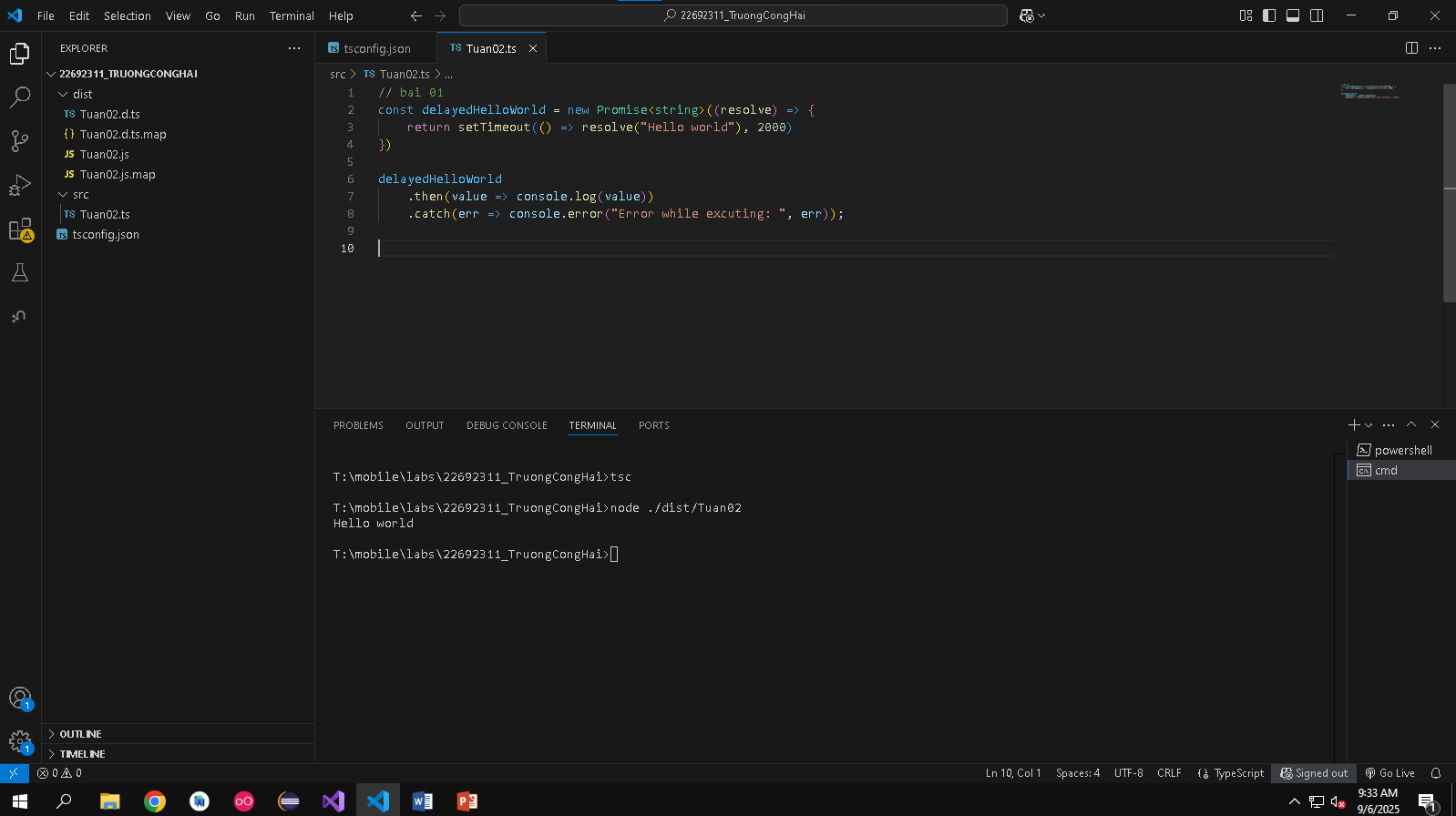
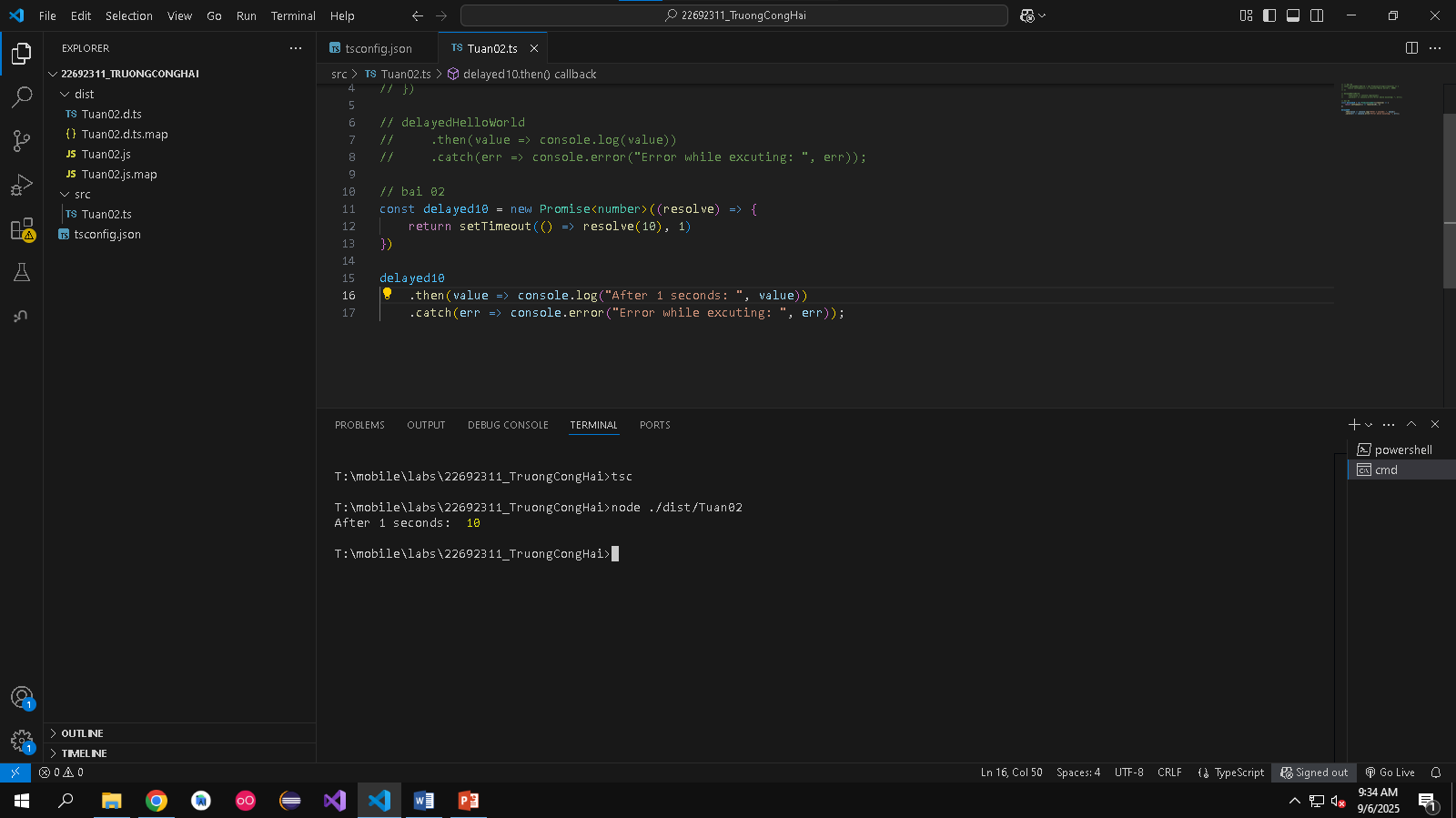
A. Basics with Promise

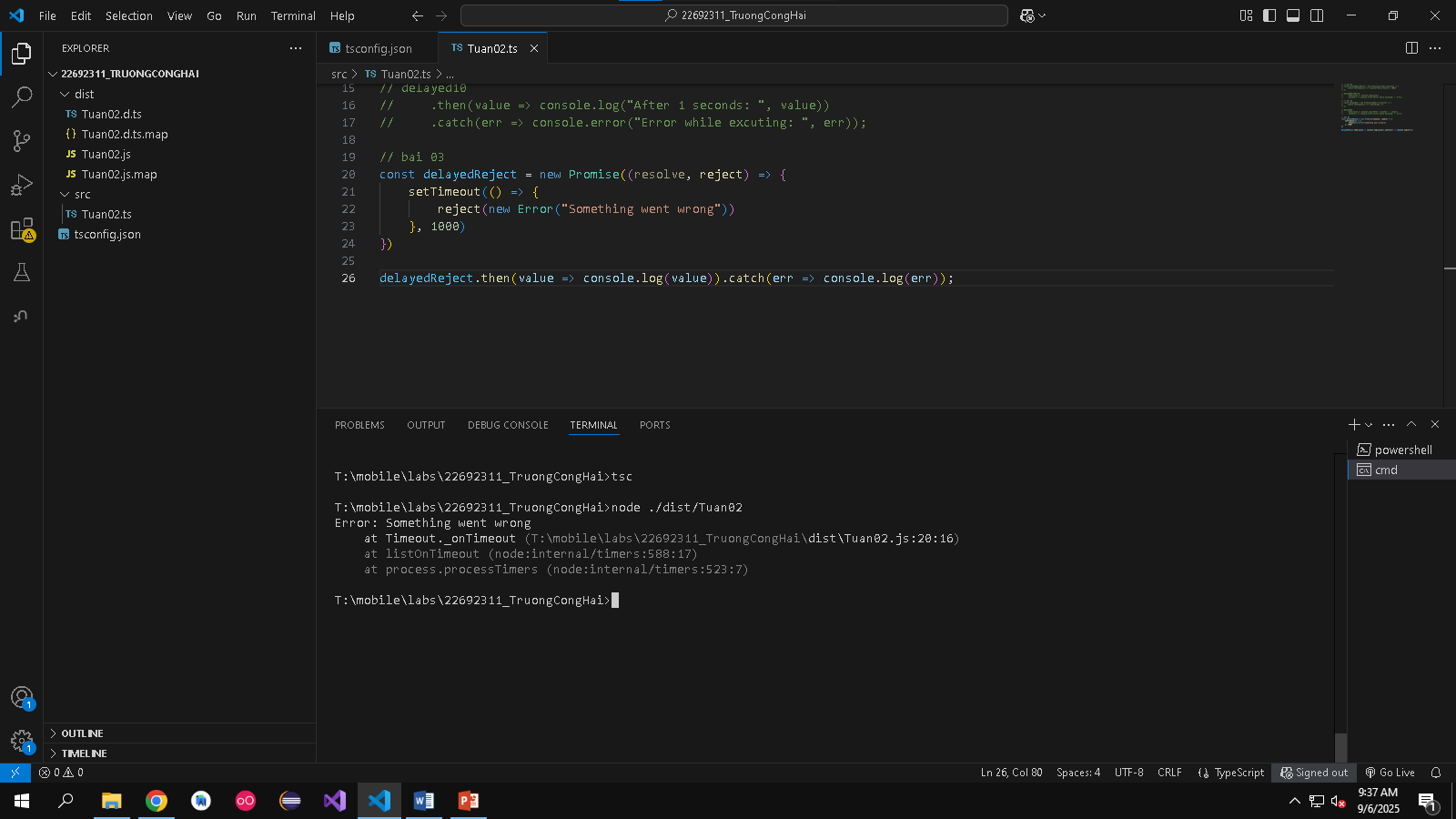
1. Create a Promise that returns the string "Hello Async" after 2 seconds.



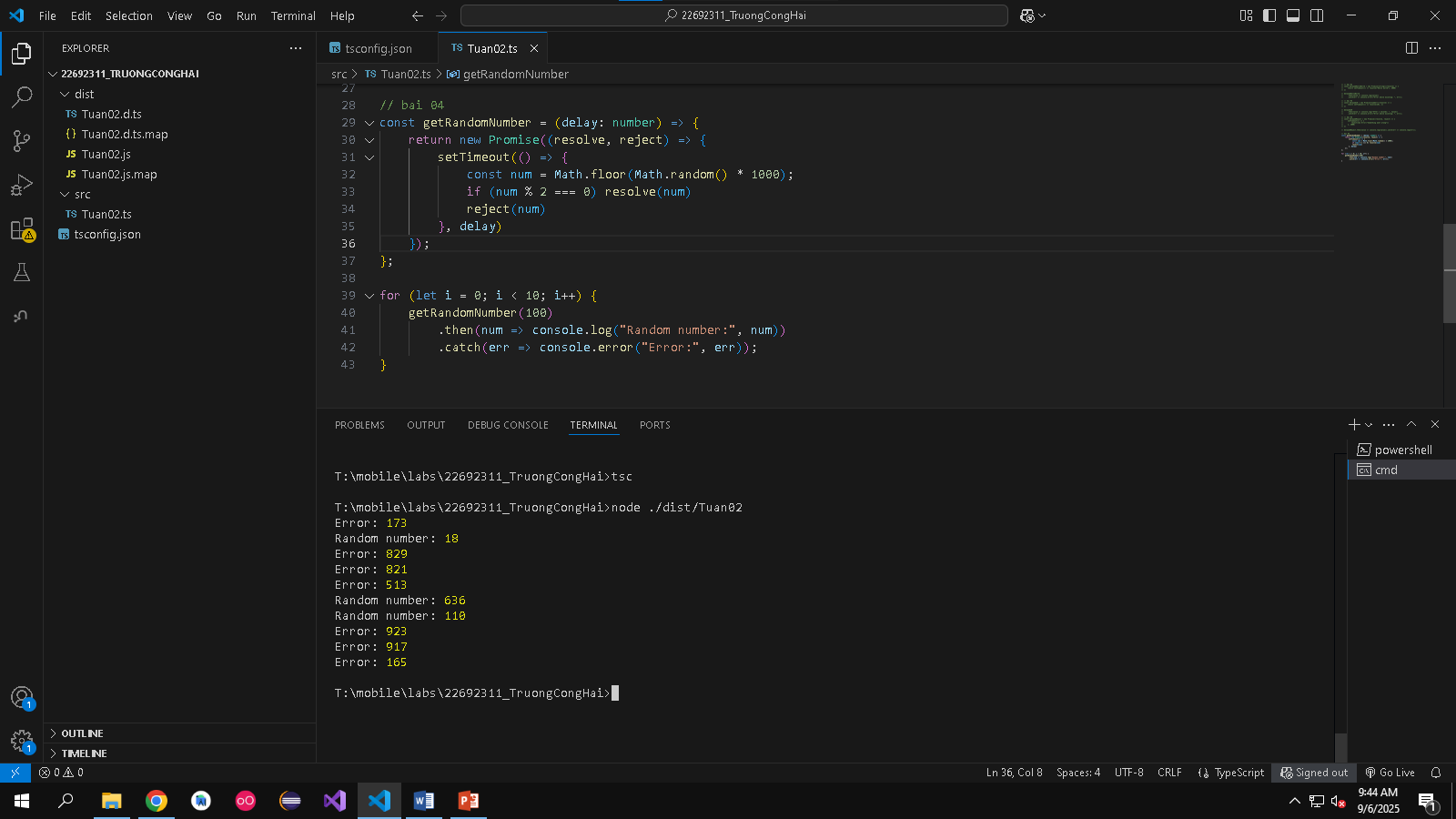
2. Write a function that returns a Promise resolving with the number 10 after 1 second.



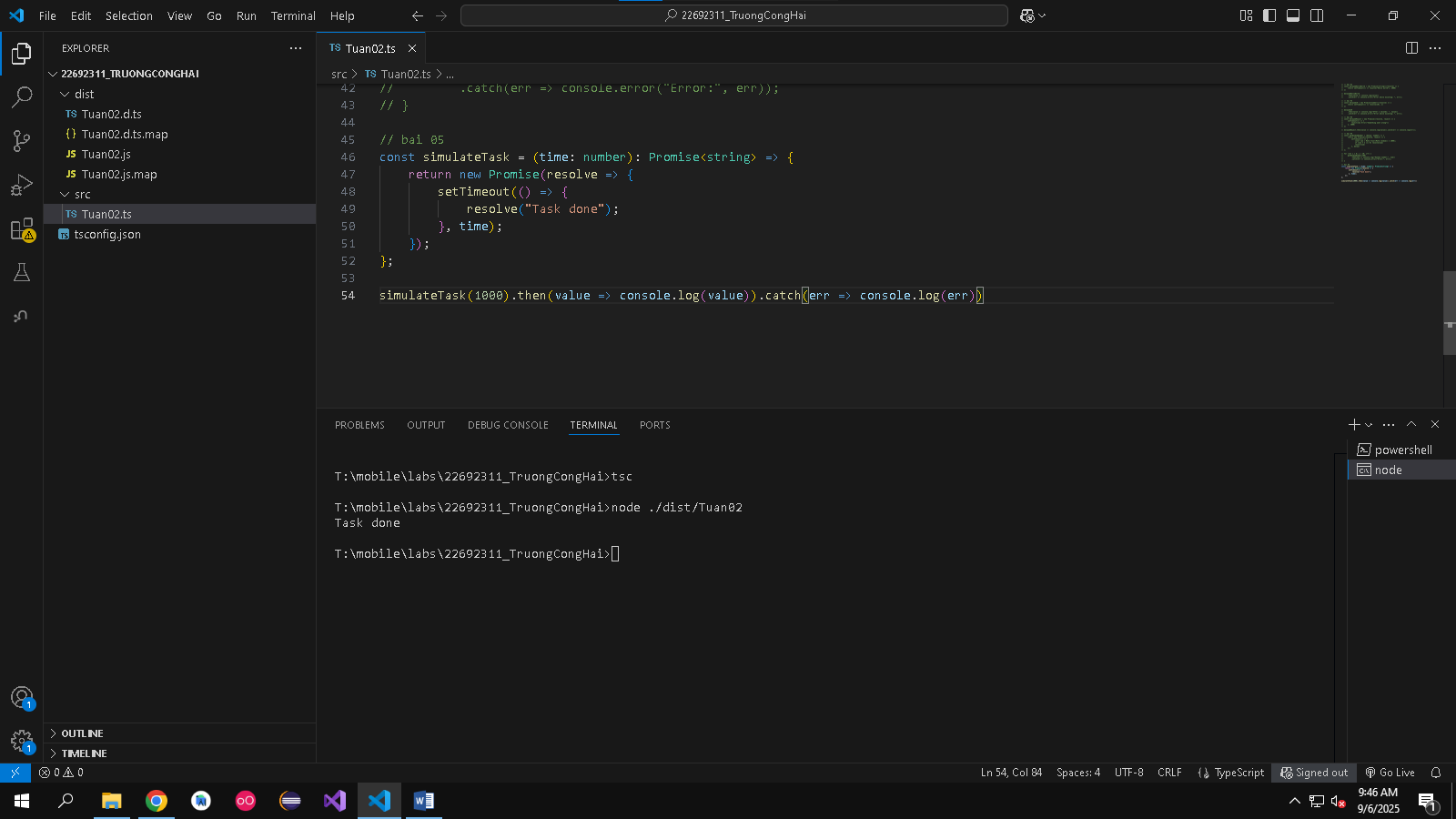
3. Write a function that rejects a Promise with the error "Something went wrong" after 1 second.



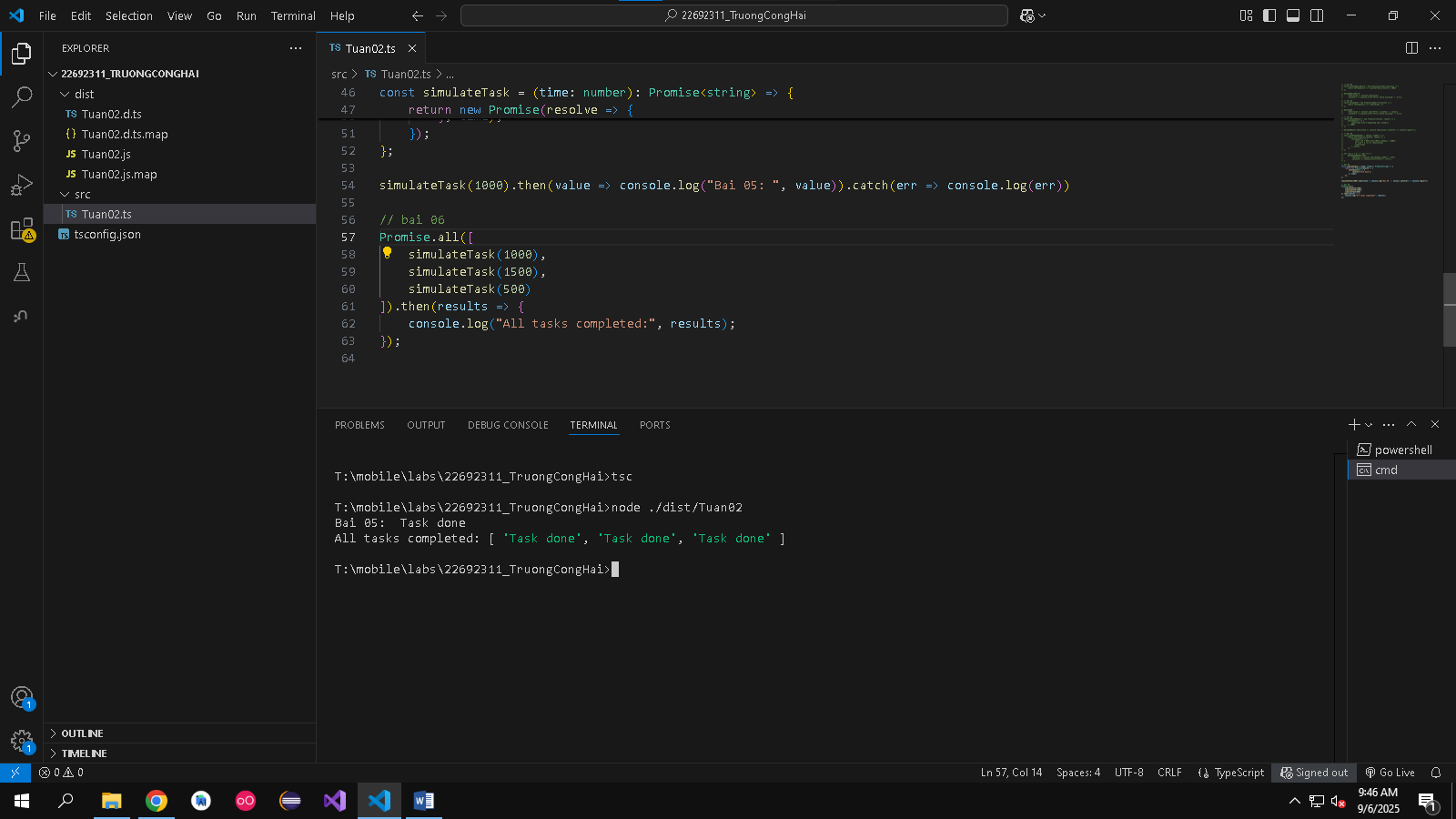
4. Use .then() and .catch() to handle a Promise that returns a random number.



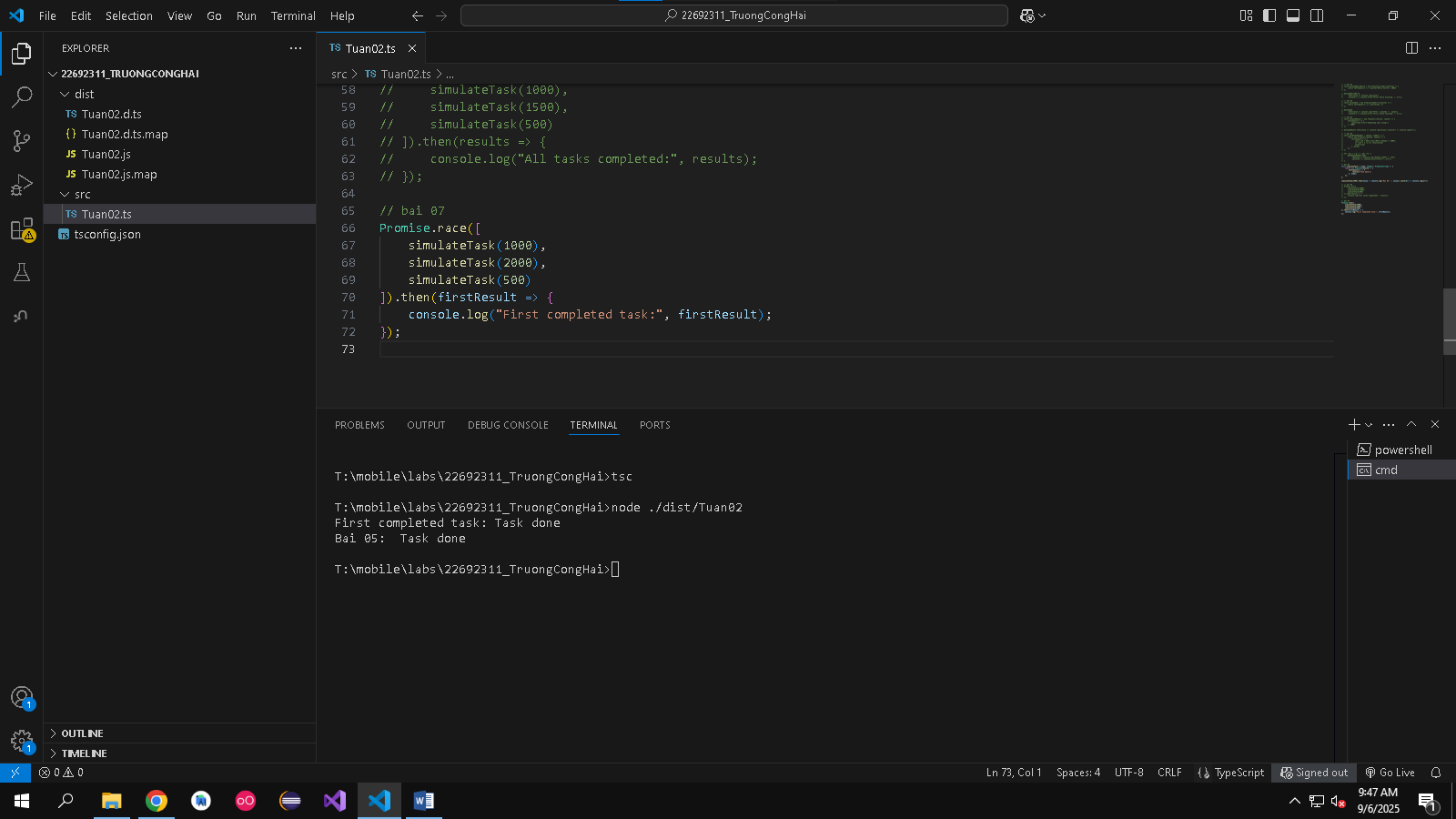
5. Create a function simulateTask(time) that returns a Promise resolving with "Task done" after time ms.



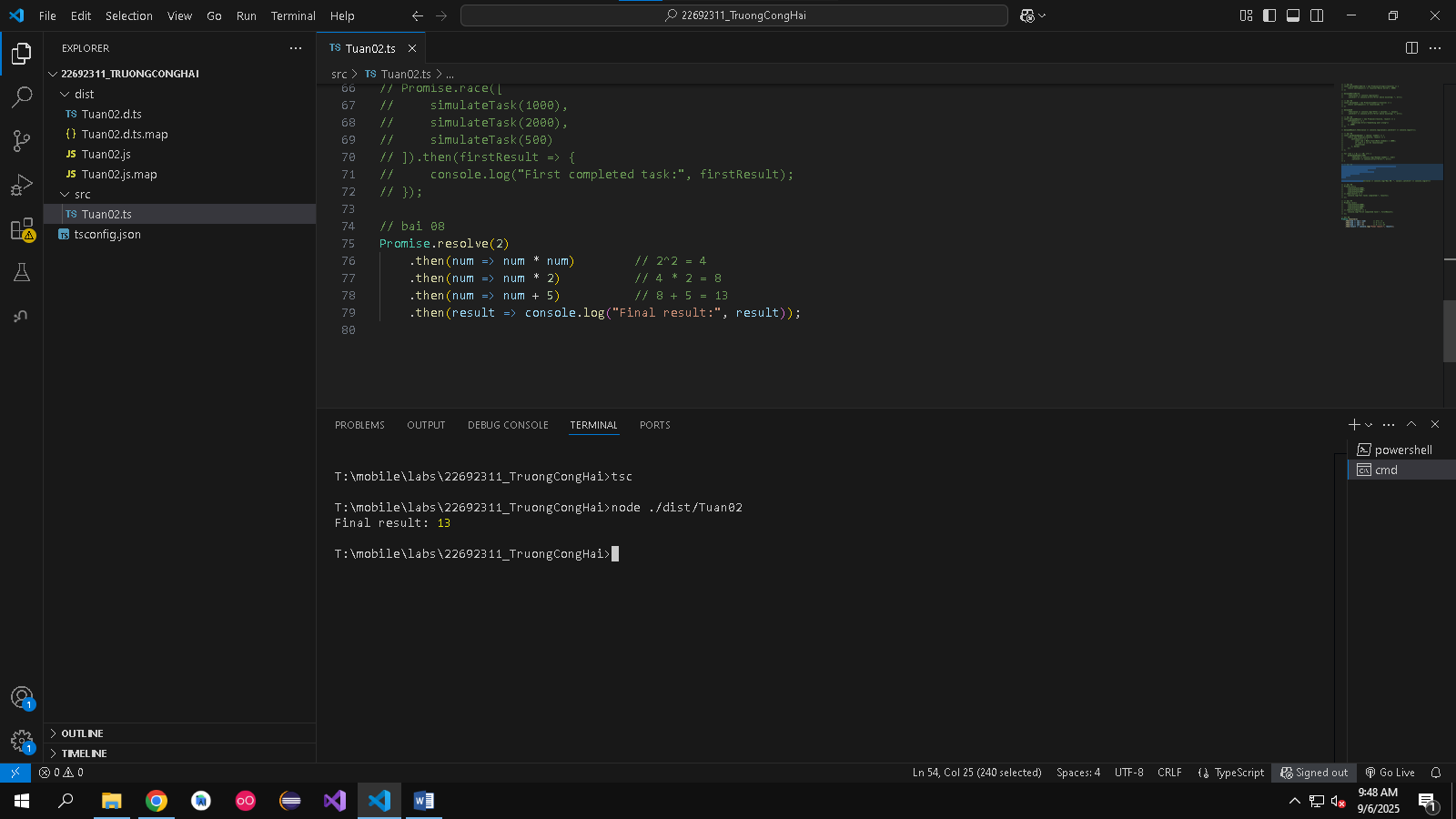
6. Use Promise.all() to run 3 simulated Promises in parallel and print the result.



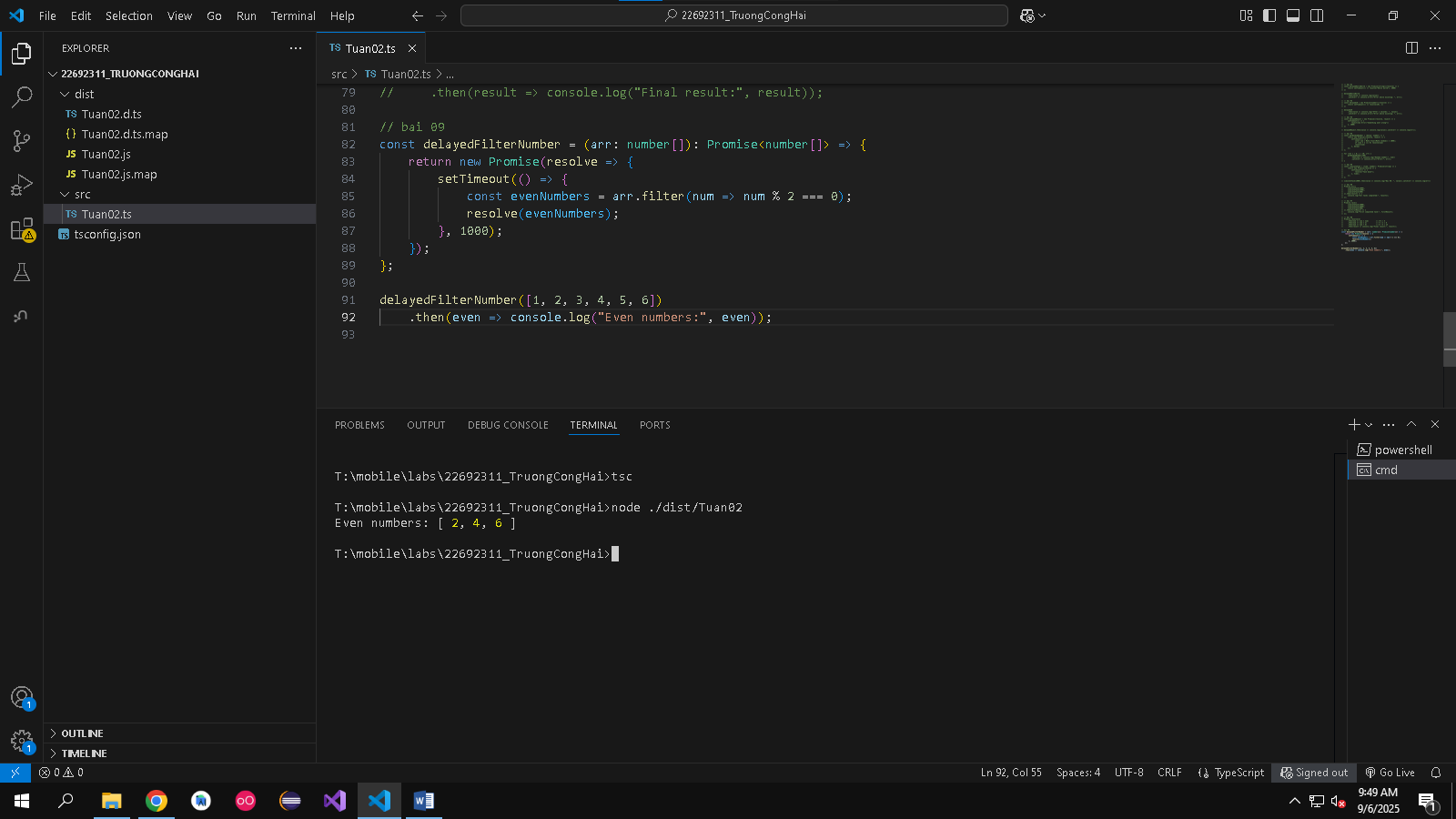
7. Use Promise.race() to return whichever Promise resolves first.



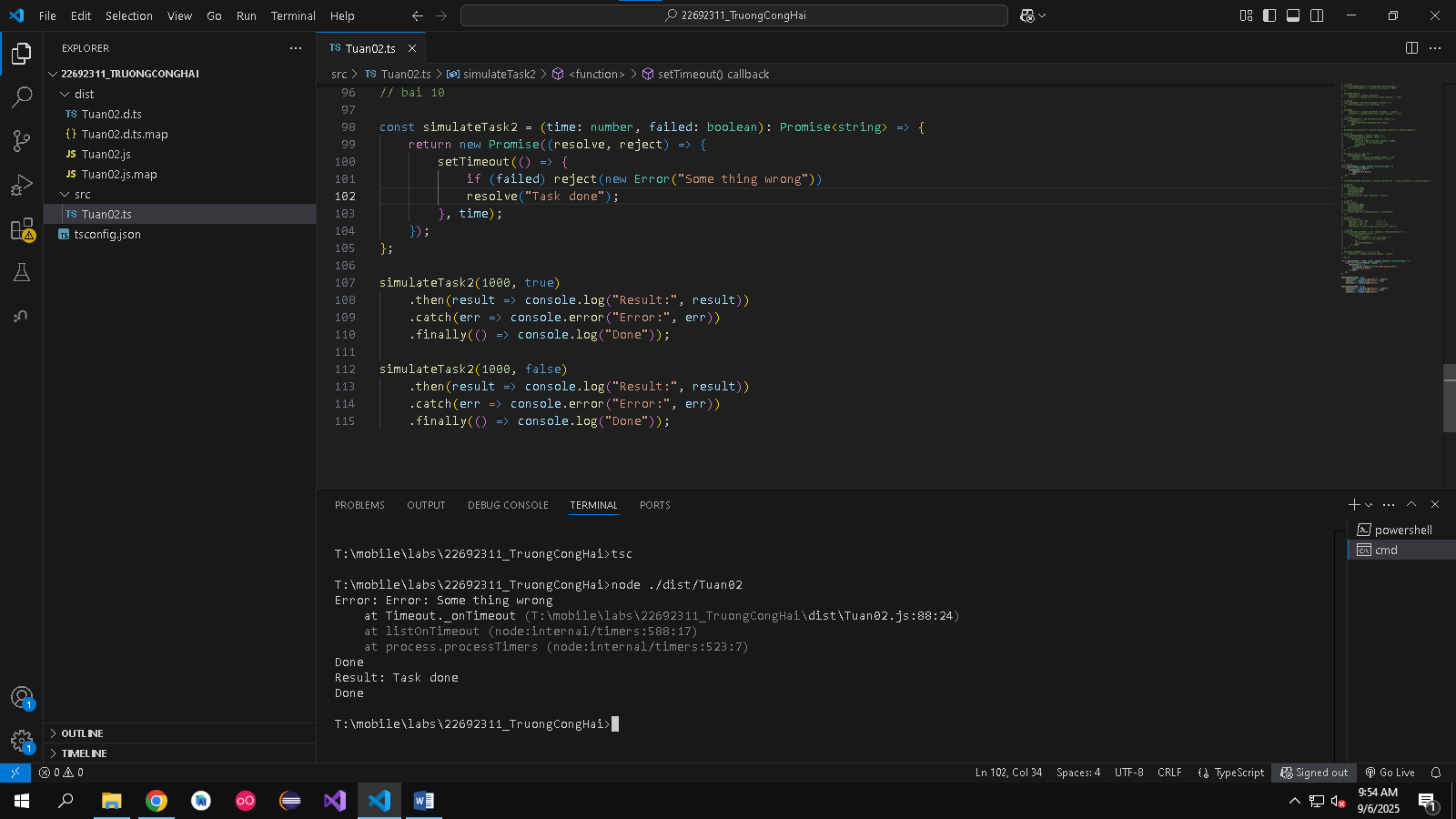
8. Create a Promise chain: square the number 2, then double it, then add 5.



9. Write a Promise that reads an array after 1 second and filters even numbers.

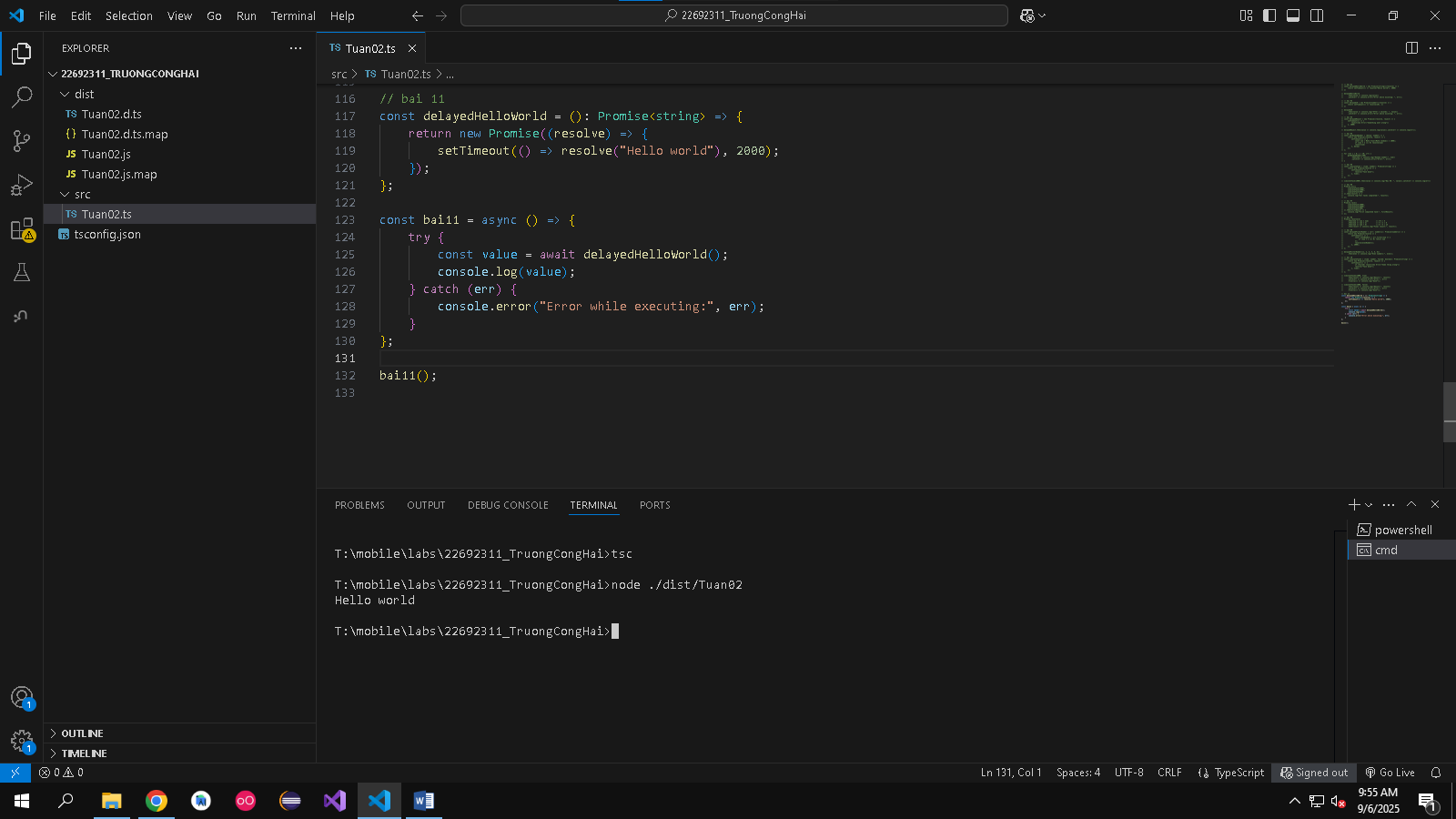


10. Use .finally() to log "Done" when a Promise finishes (success or failure).

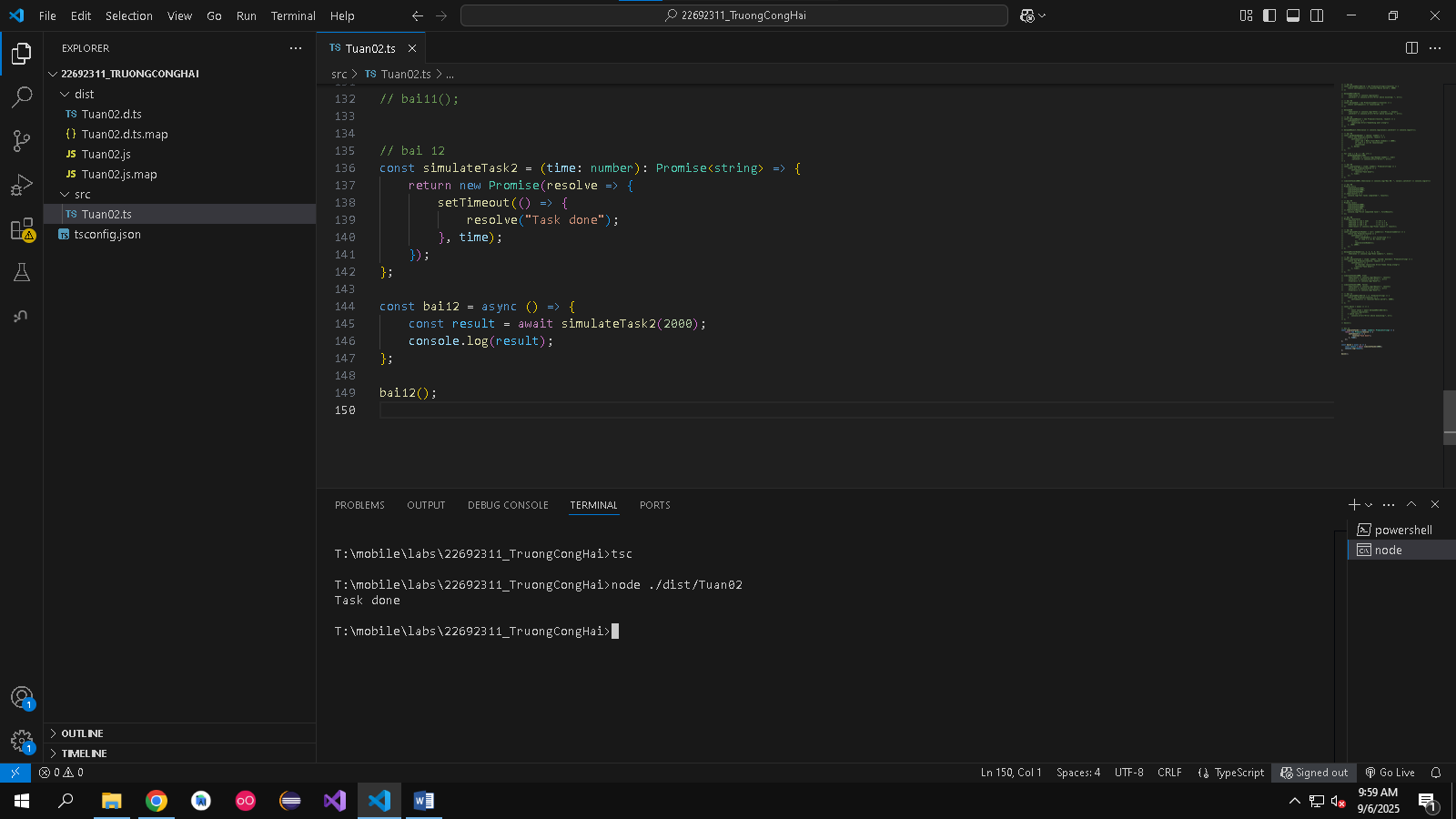


B. Async/Await

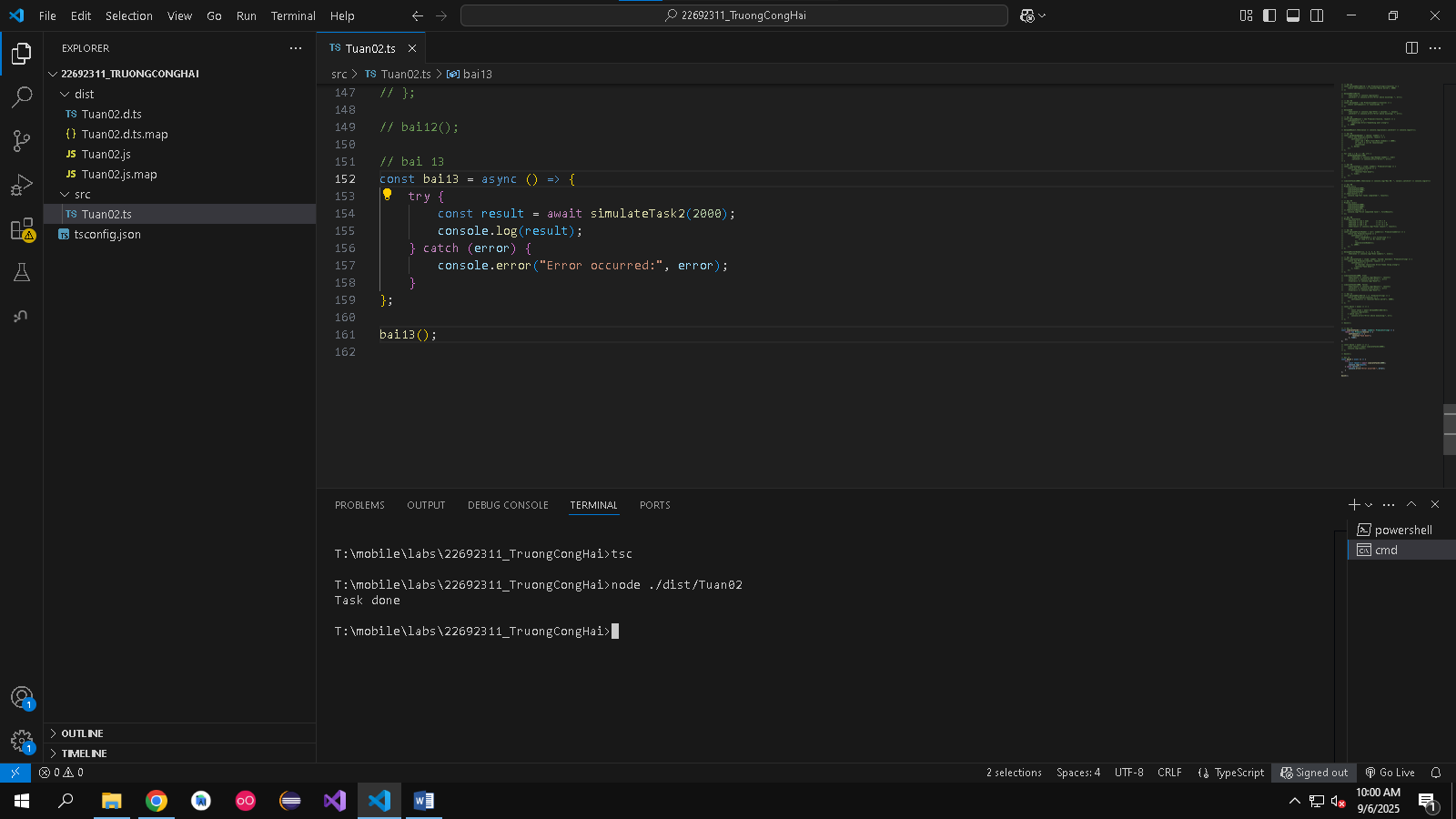
11. Convert Exercise 1 into async/await.



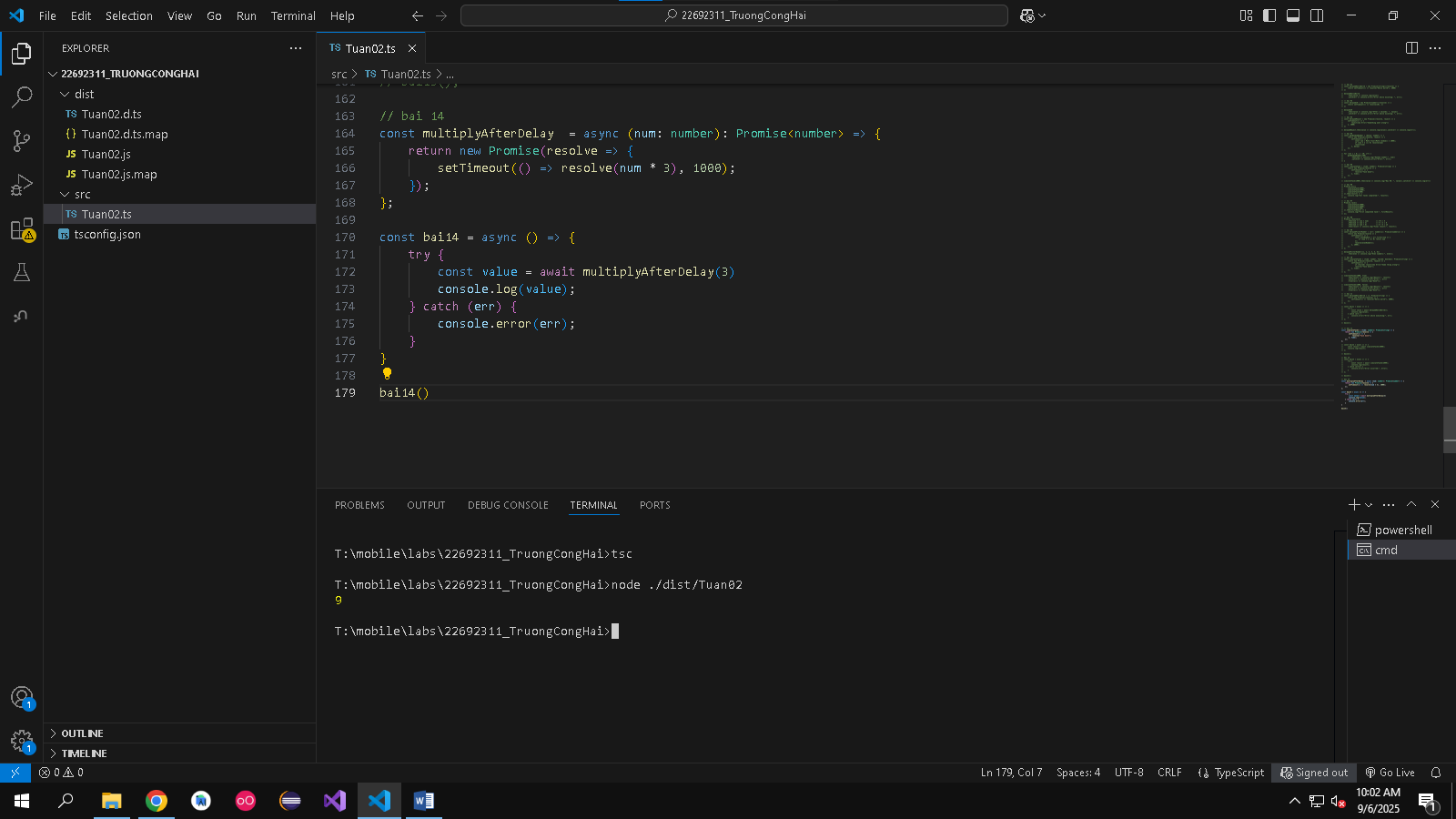
12. Write an async function that calls simulateTask(2000) and logs the result.



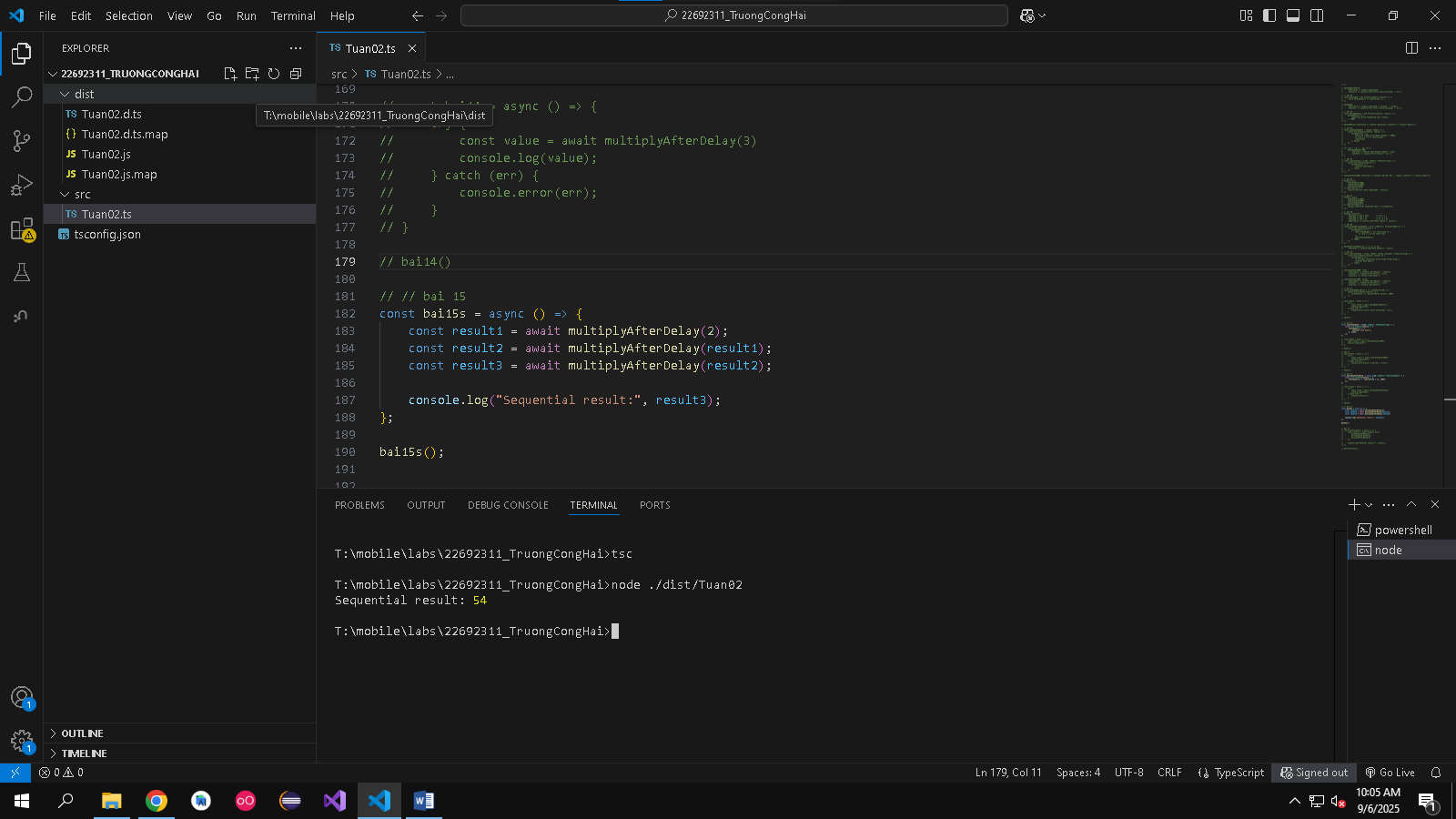
13. Handle errors using try/catch with async/await.



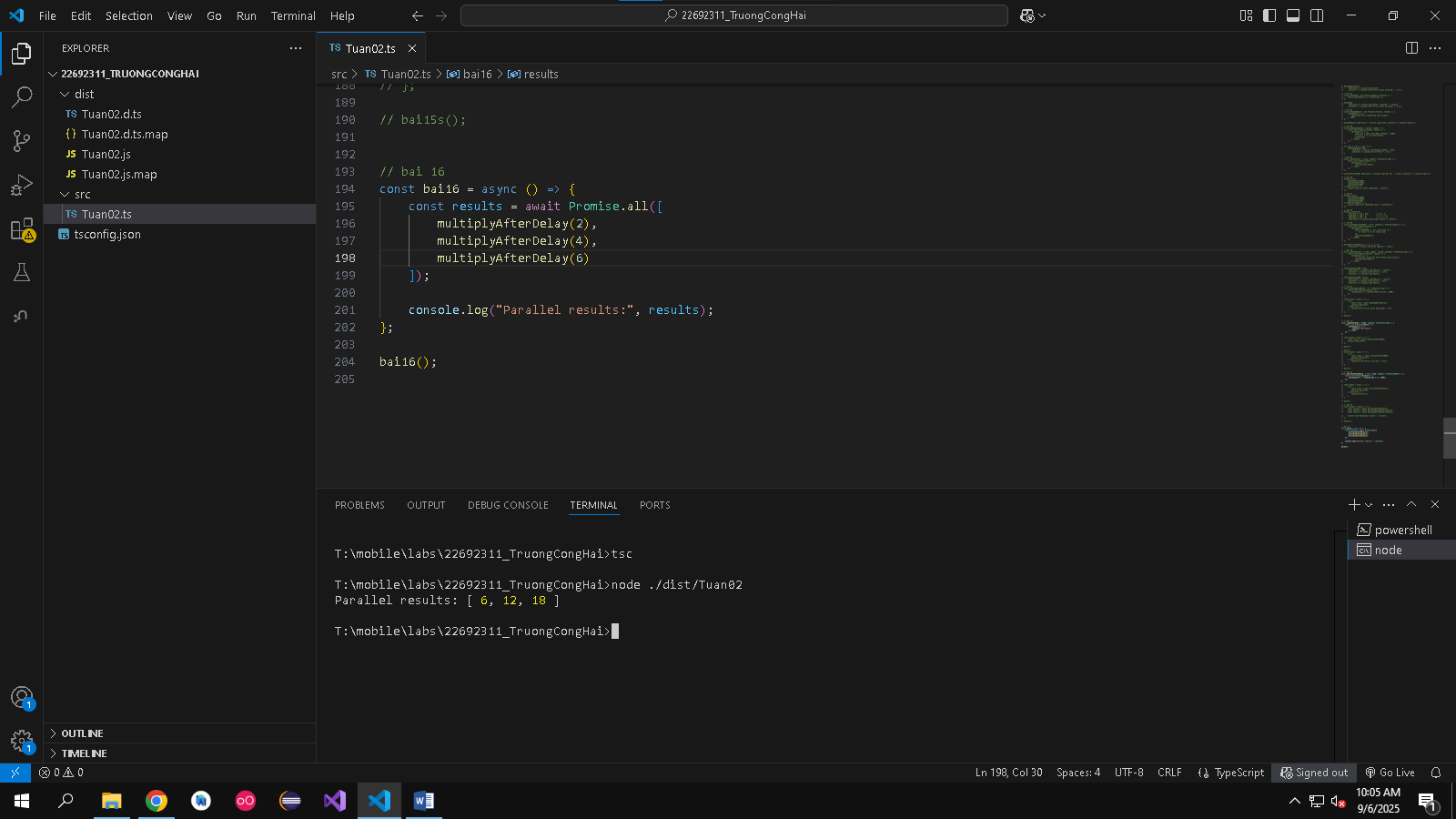
14. Write an async function that takes a number, waits 1 second, and returns the number × 3.



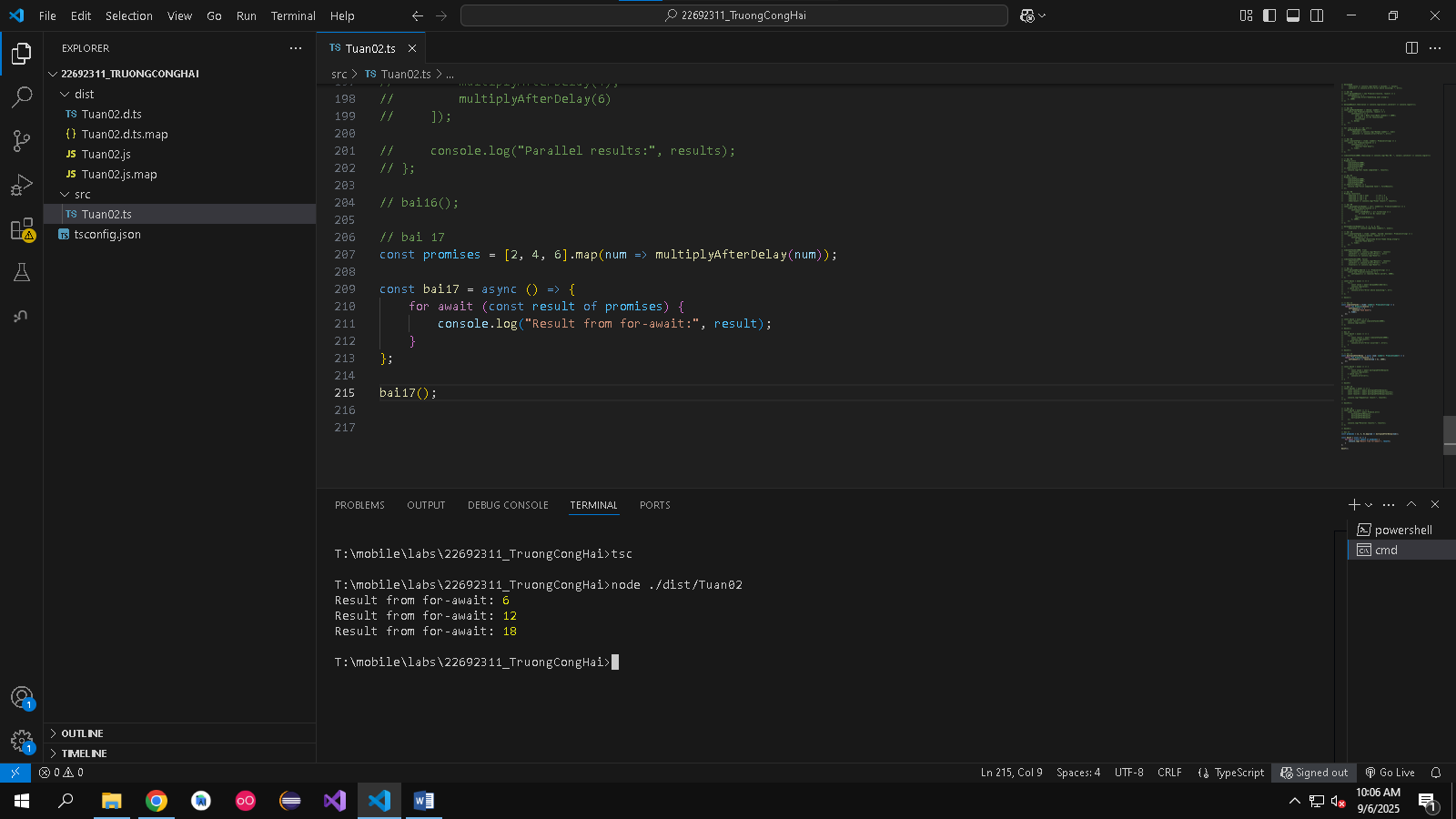
15. Call multiple async functions sequentially using await.



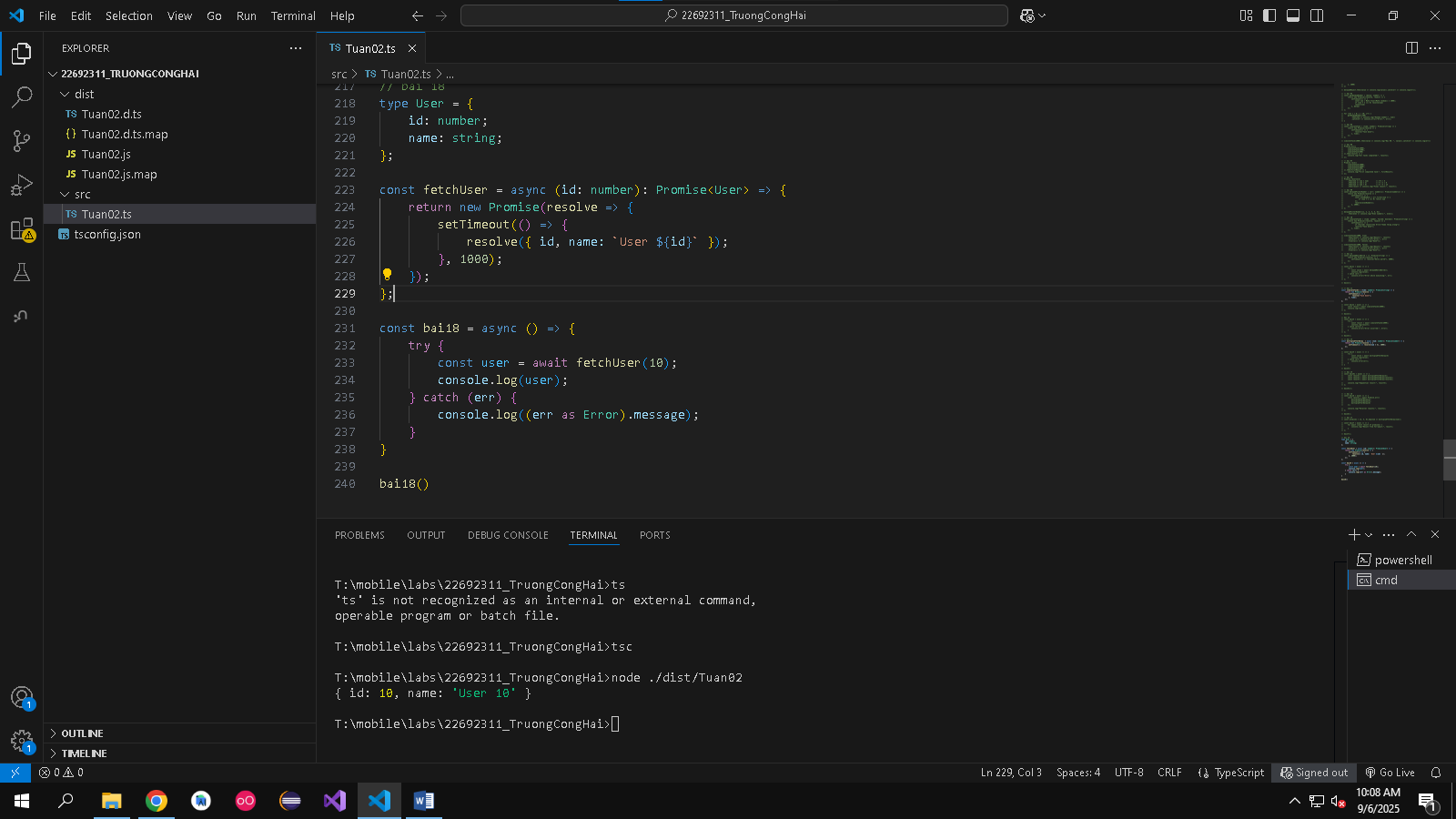
16. Call multiple async functions in parallel using Promise.all().



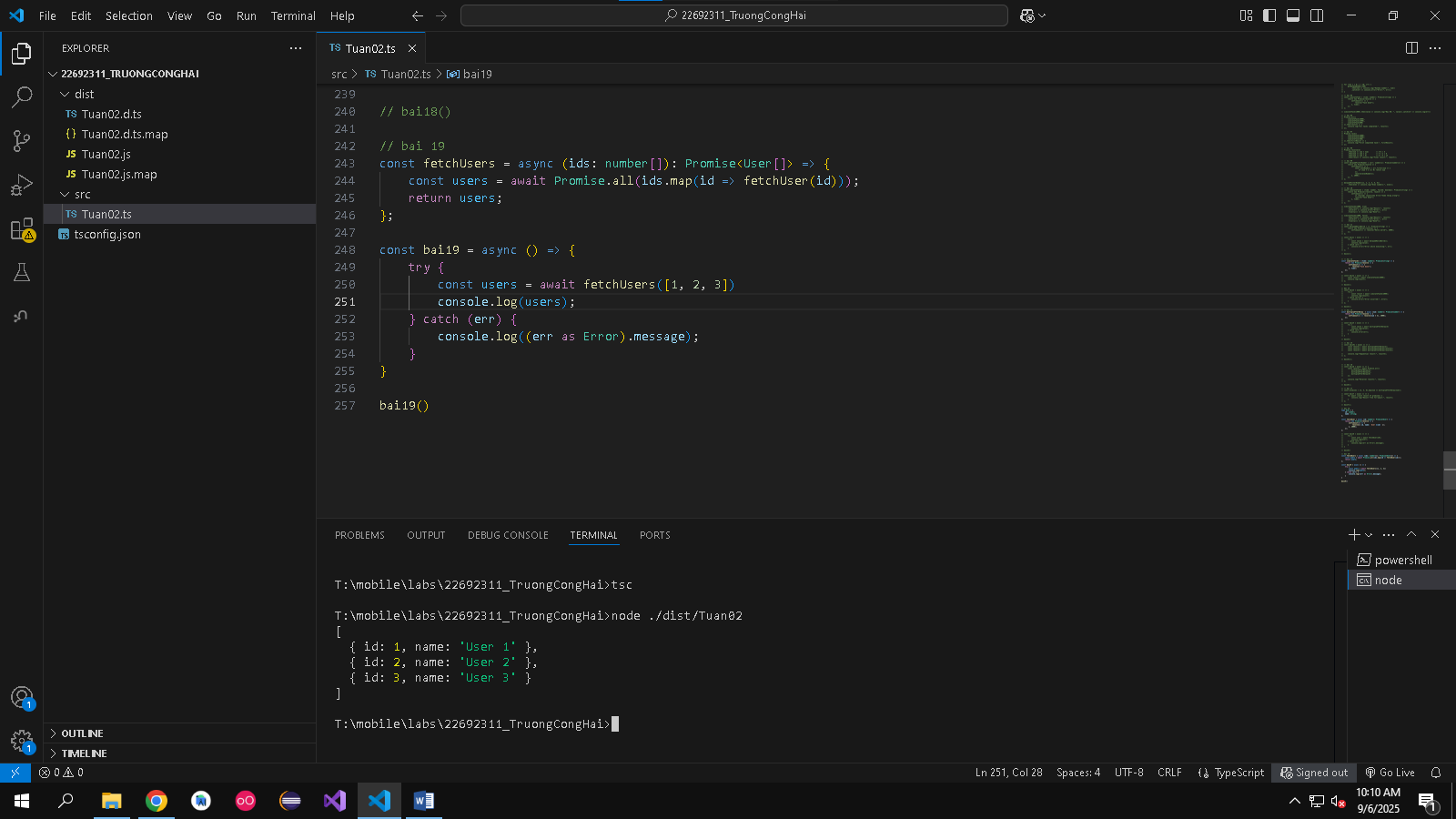
17. Use for await...of to iterate over an array of Promises.



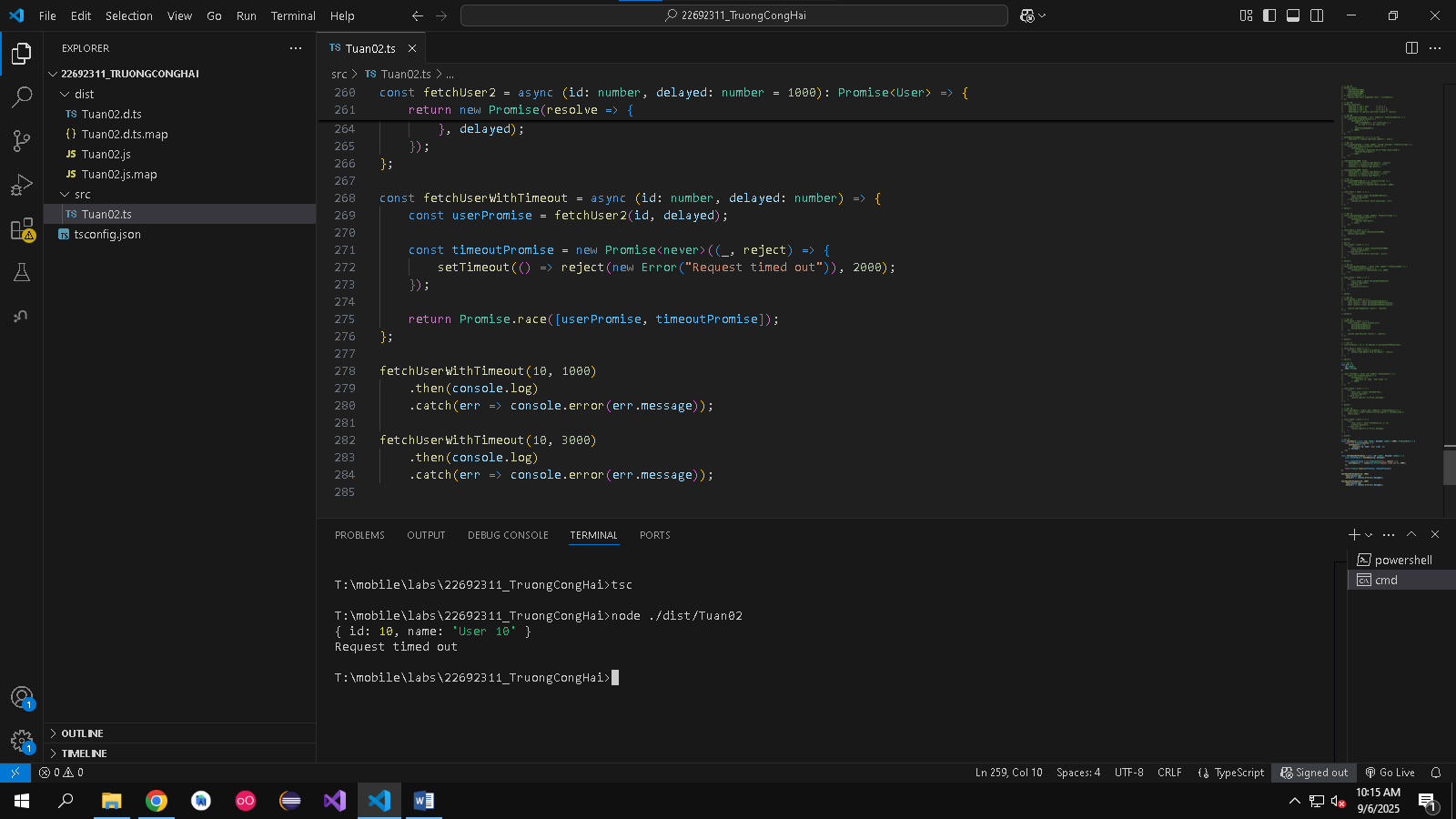
18. Write an async function fetchUser(id) that simulates an API call (resolves a user object after 1 second).



19. Create an async function fetchUsers(ids: number[]) that calls fetchUser for each ID.



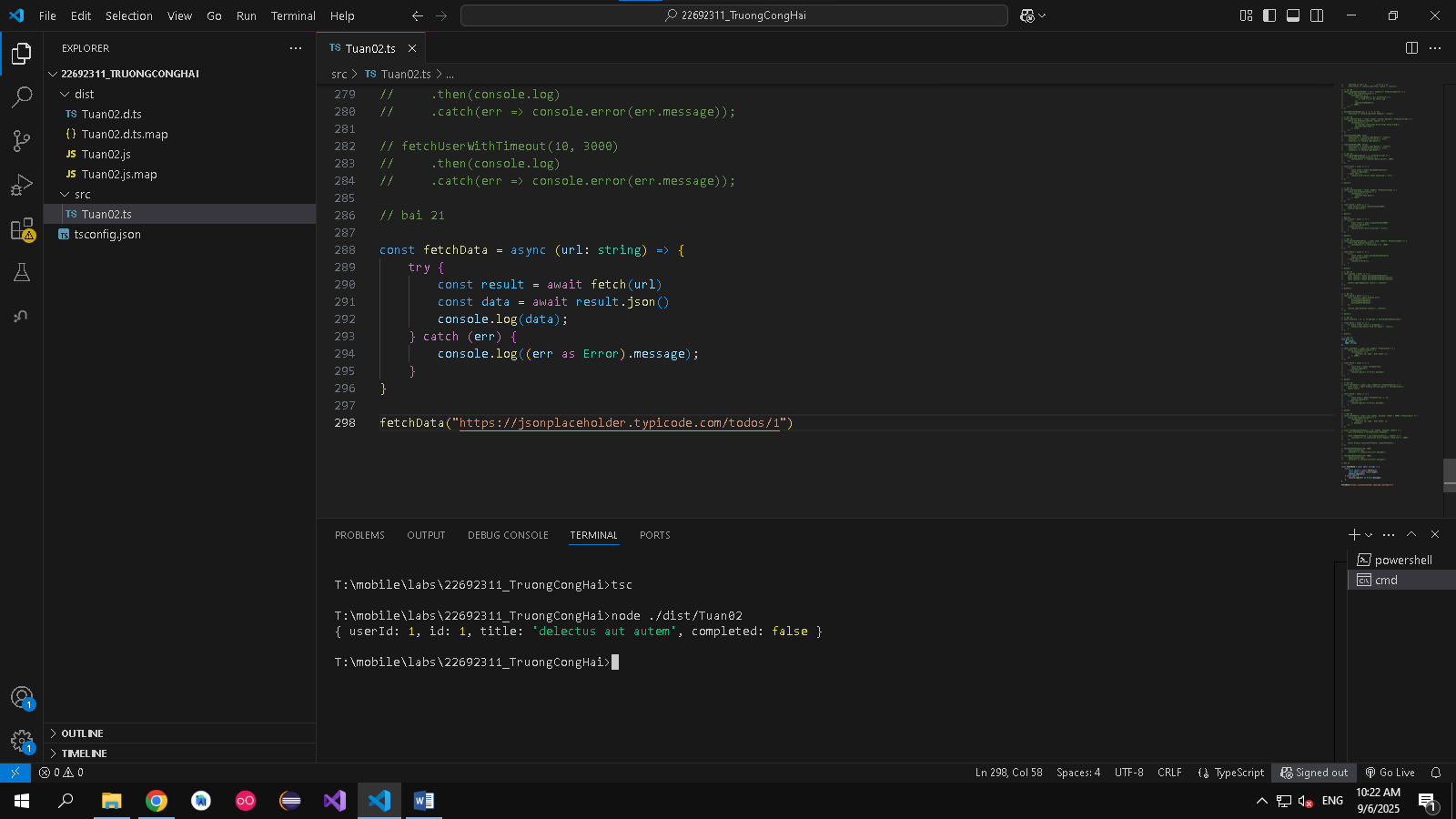
20. Add a timeout: if the API call takes more than 2 seconds, throw an error.



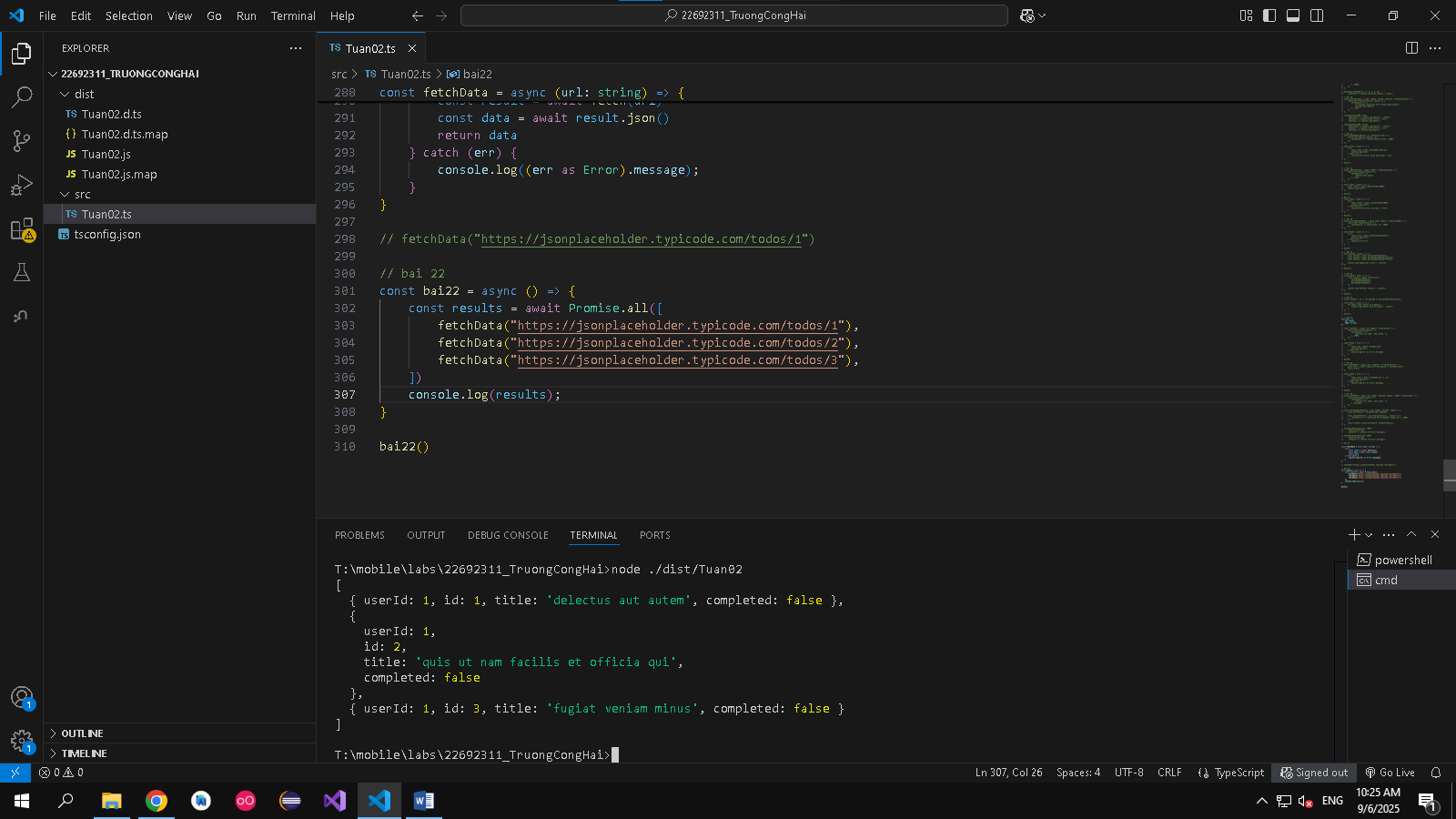
Dùng Promise.race để tìm ra promise nào trả về sớm nhất giữa fetchUser và timeOut trong 2s, nếu fetchUser trả về trước timeOut thì không bị lỗi.

C. Fetch API & Simulated I/O

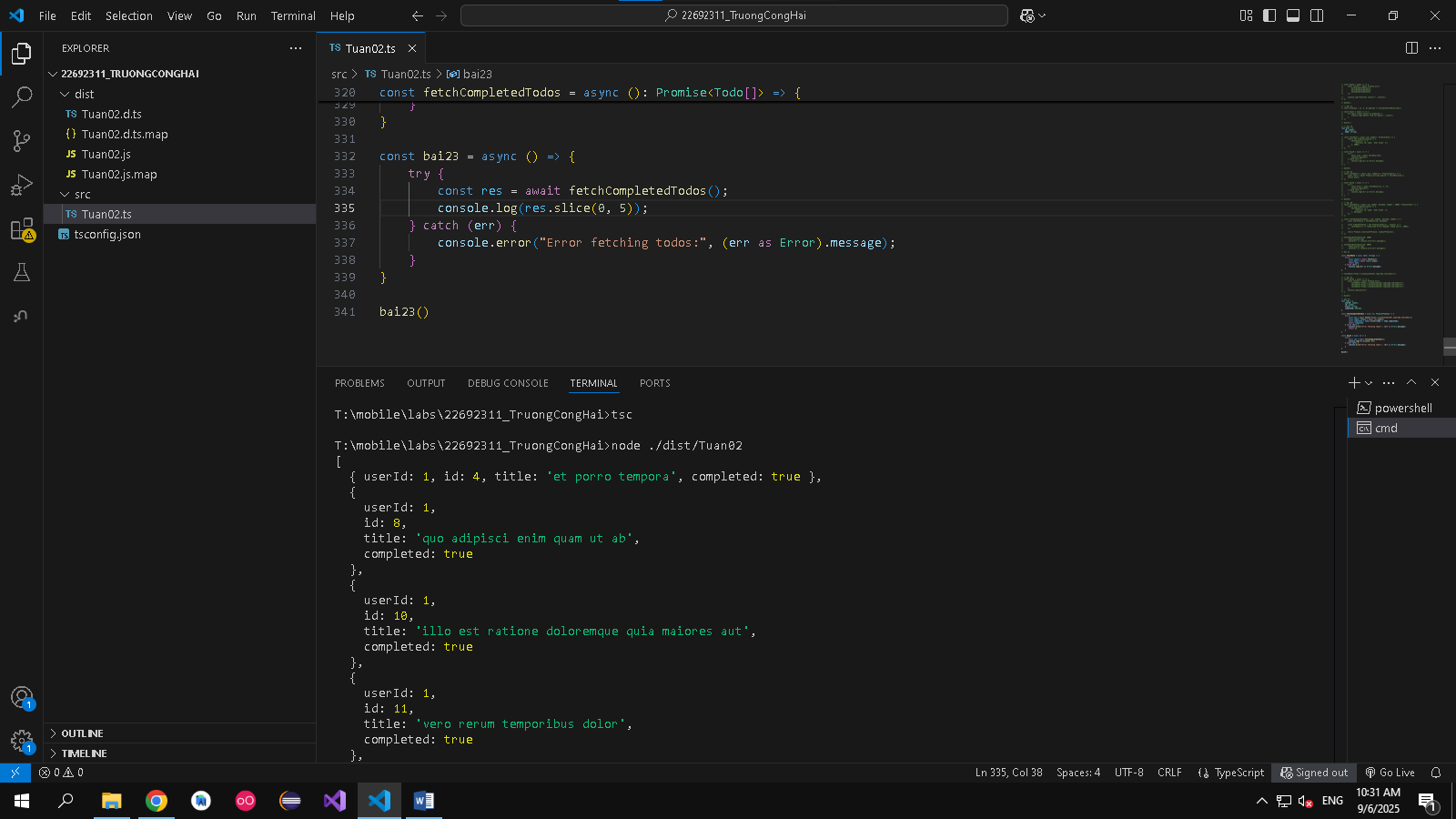
21. Use fetch to get data from a public API (e.g., <https://jsonplaceholder.typicode.com/todos/1>).



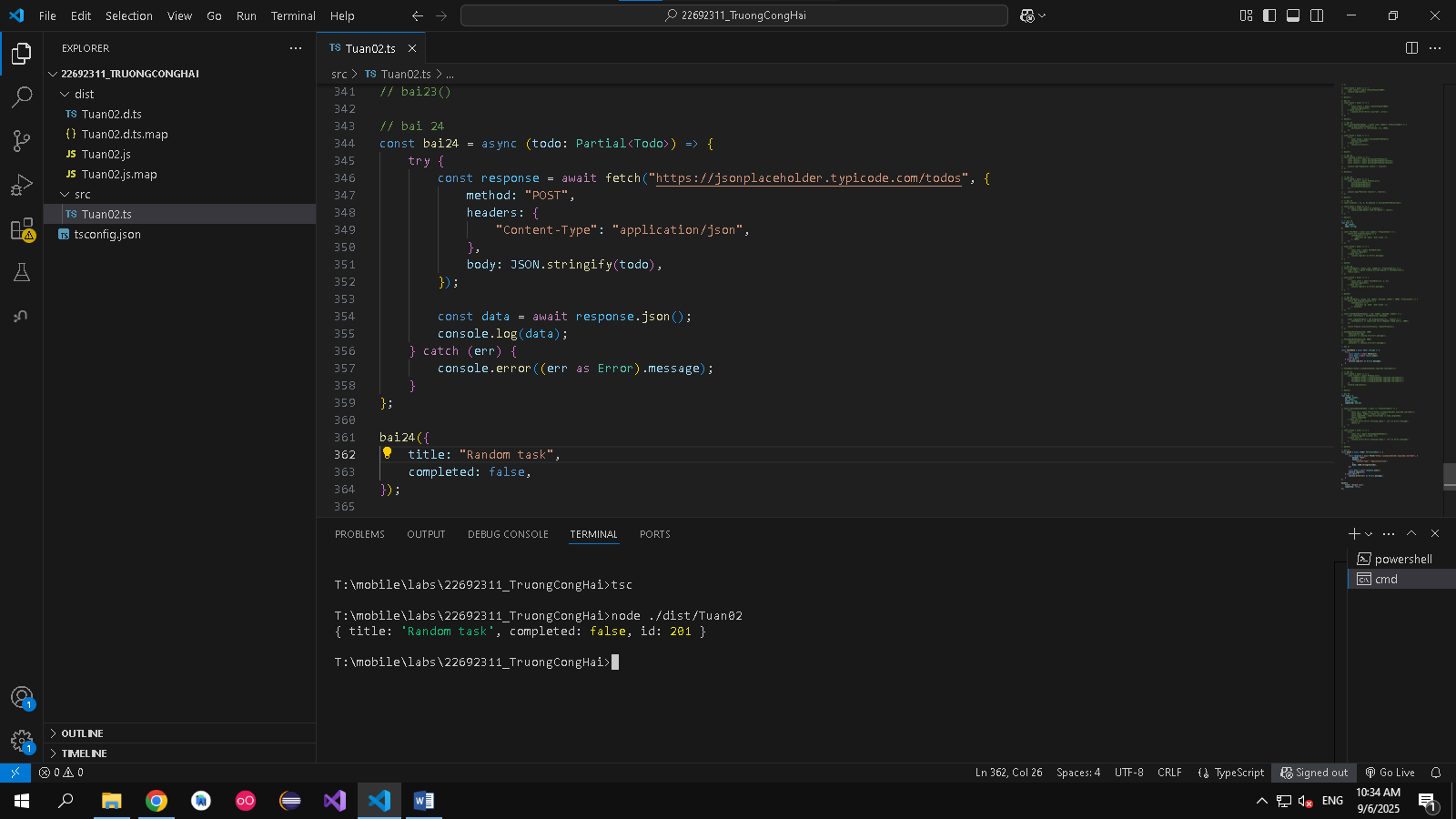
22. Call the API multiple times and log the results.



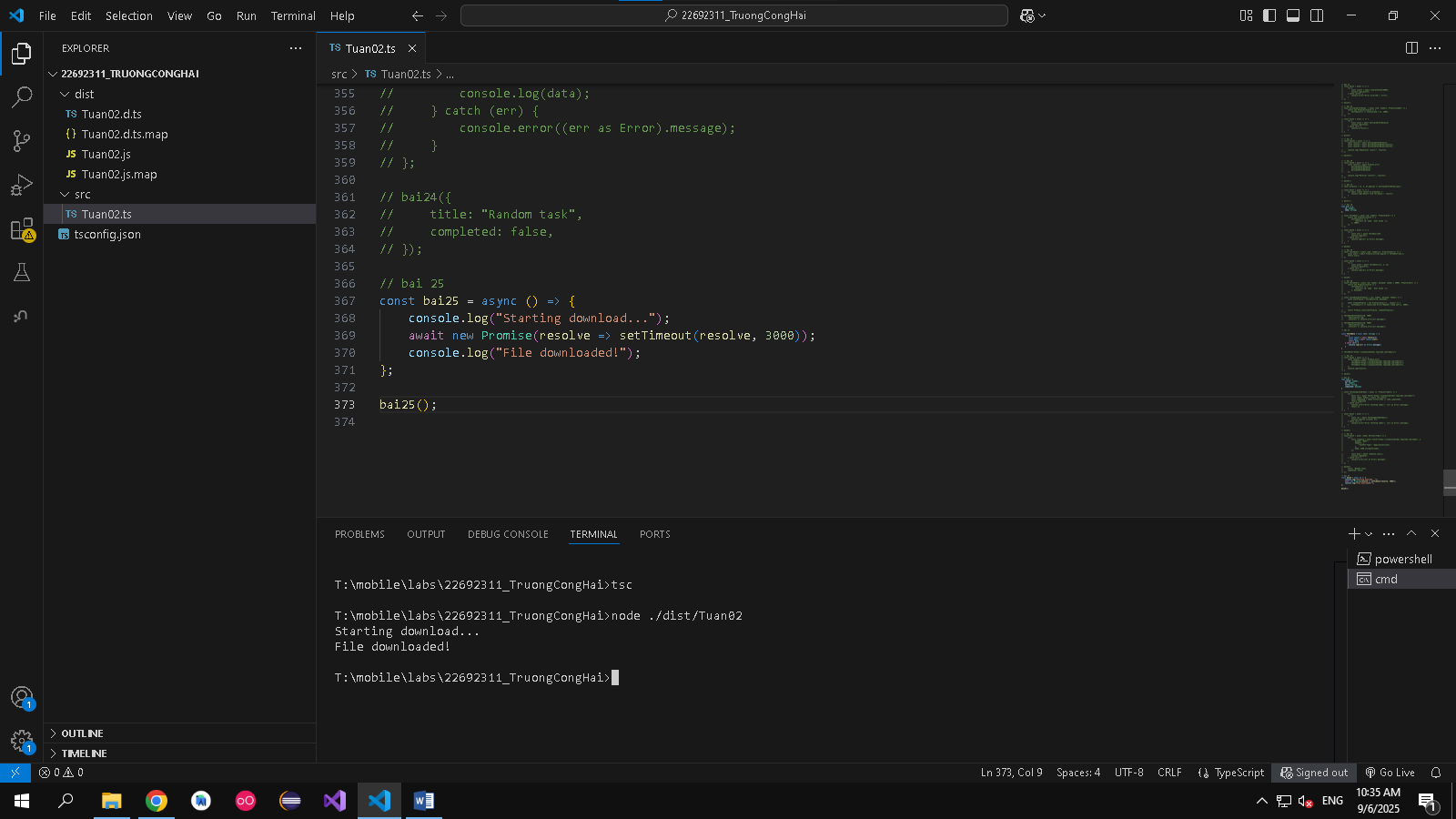
23. Write an async function that fetches a list of todos and filters out those that are not completed.



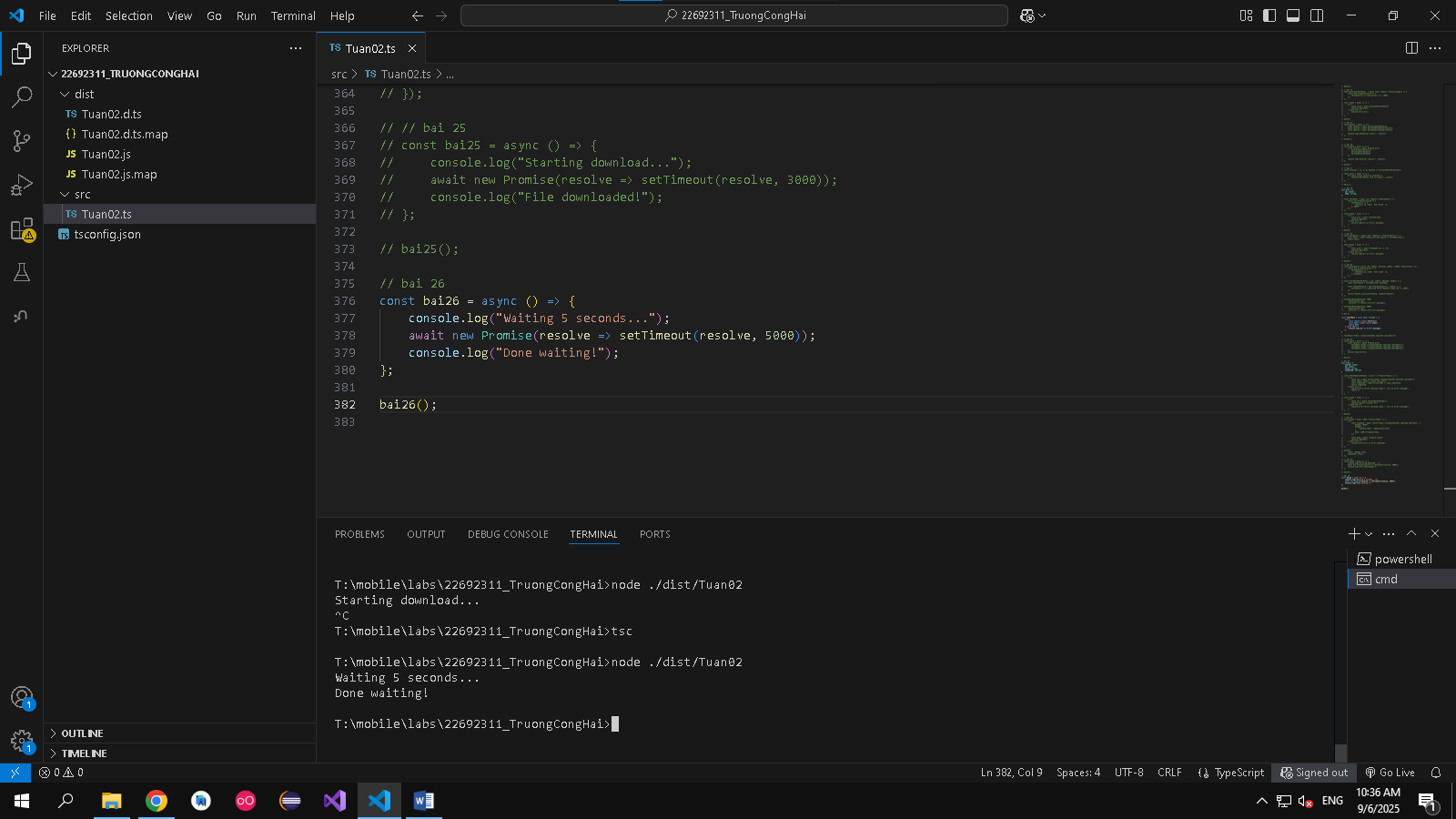
24. Write an async function postData() that sends a POST request to a test API.



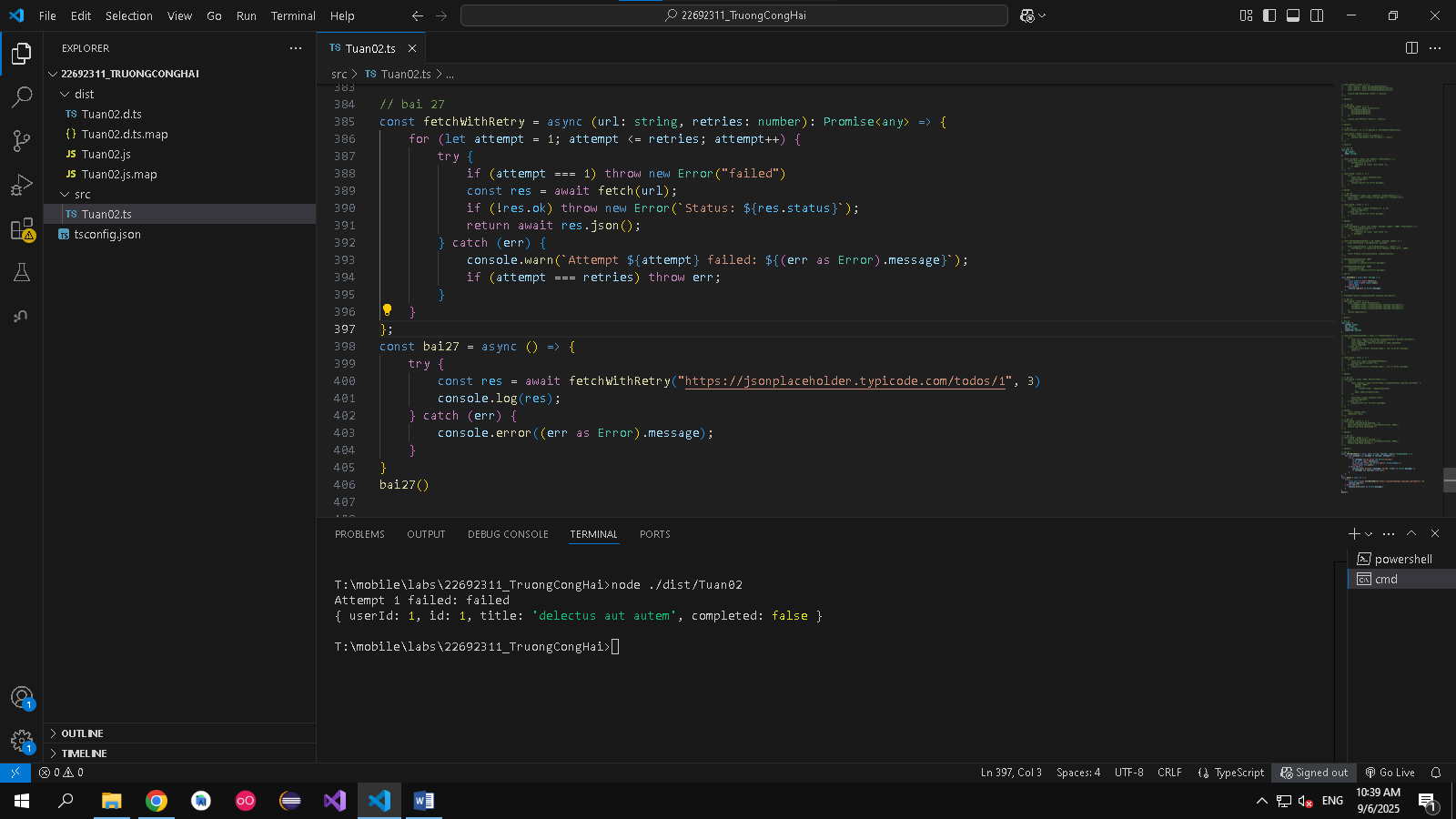
25. Create a function downloadFile that simulates downloading a file in 3 seconds and logs when done.



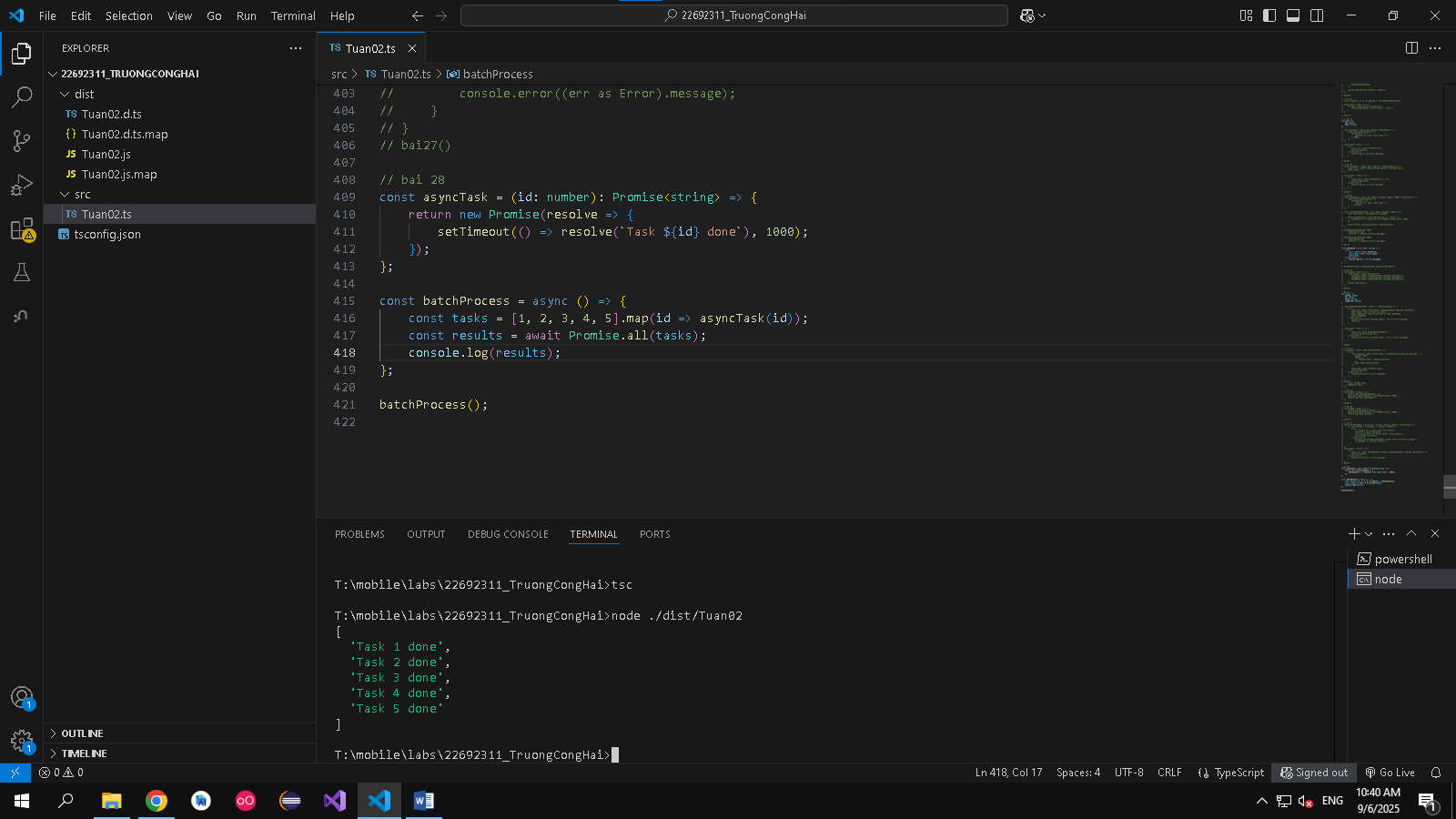
26. Use async/await with setTimeout to simulate a 5-second wait.



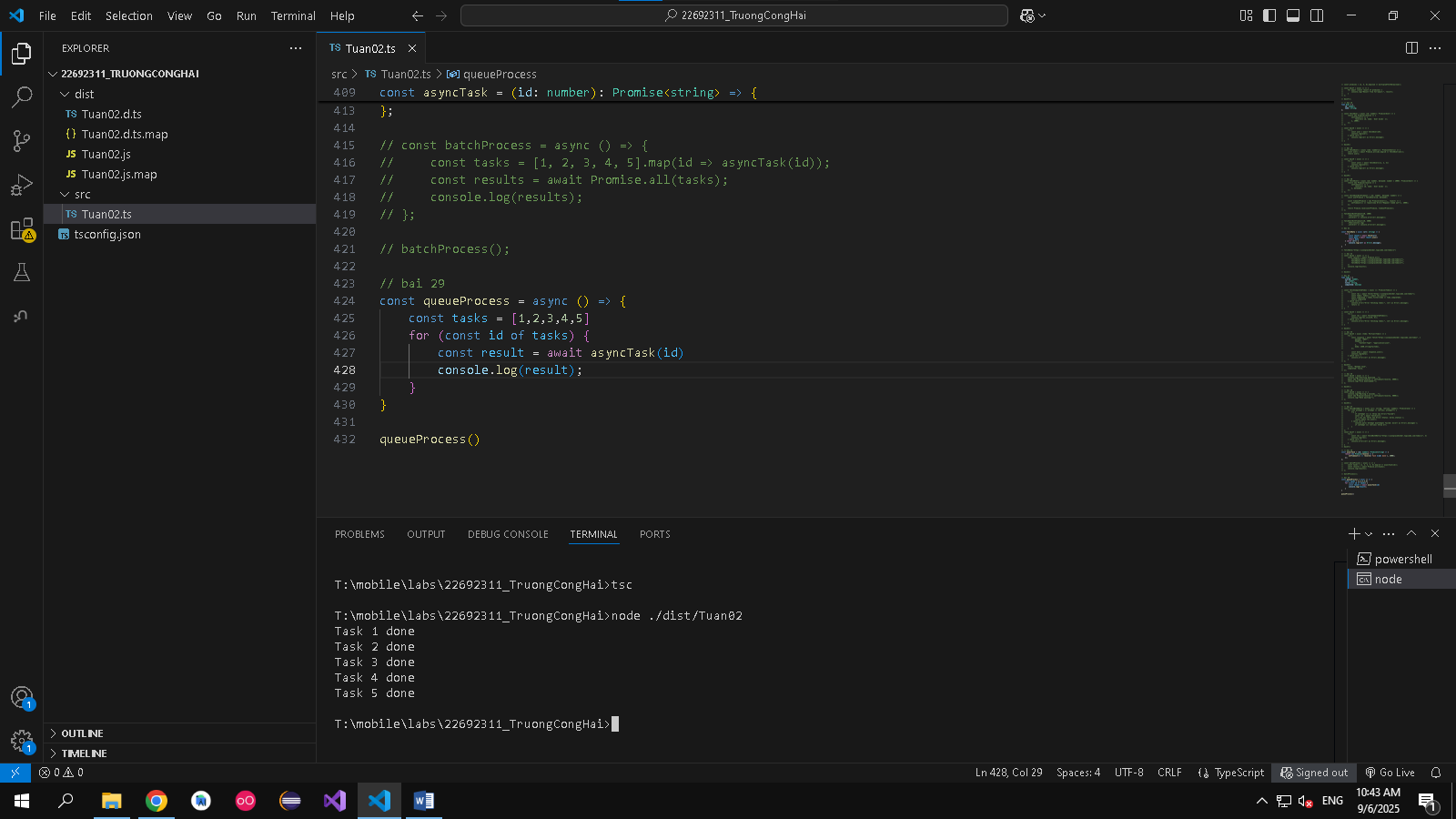
27. Write a function fetchWithRetry(url, retries) that retries up to retries times if the API call fails.



28. Write an async function batchProcess() that processes 5 async tasks at once (use Promise.all).



29. Write an async function queueProcess() that processes tasks sequentially in a queue.



30. Use async/await + Promise.allSettled() to handle multiple API calls and display their success/failure status

