Testing Approach:

1. Tech Deck: Using C# + Playwright + Nunit
2. Using page object model
3. Four different categories to be run at different scenarios and setup.
    a. Smoke - Smoke tests suite that are not dependent on database/API access or presetup data, can be run in any environment.
    b. E2E - Tests that mimic whole use cases (workflow), may have dependency on data setup, can be run in staging and lower environments.
    c. Regression - Tests suite that are more focused on covering all use cases and models, can be run at Alpha and Beta environments.
    d. Functional - Tests suite that including positive and negative functional tests, run at alpha (lower testing) environments.
4. AppSettings.json to store all the configurations, password and other secure data can be stored at the secret manager at AWS.
5. Memo on test cases:
    a. Focus on the input fields: investment amount and duration slider
    b. Assume validation check will be taken on the investment amount textfield once out of focus.
    c. For regression and functional tests, if have access to database or API, should validate the data on UI with the real data stored.

Performance Testing:

1. Baseline threshold testing can be embedded into the regression test suite.
    a. Talk to PM and devs, get all the thresholds (page loading/redirecting max time, purchase submit response max time, etc.)
    b. Timeout and fail the test once exceeds threshold
    c. Record these performance data into a database for the recent three to five releases, generate reports and highlight sudden performance drops even not exceeding threshold.
2. Performance - load testing and disaster recovery testing can be performed by a performance testing tool for better vision and results (like Gutling or JMeter)
    a. Environment: Prod or Prod-like environment (for example: with load balancer and HA databases)
    b. Monitor:
        i. CPU/memory on all web servers, backend processors and attributes for databases if any
        ii. Logs on hits, errors and performance threshold.
        iii. UI response time
        iv. Backend/database actions response time if any
    c. Test Scenarios:
        i. Large number of users load the page at the same time.
        ii. Large number of users hit the purchase button at the same time.
        iii. Some of the webservers are not available
        iv. Some of the databases are not available

v. Some of the backend processors are not available if any.

d. Generate reports and discuss findings with related parties.