Cypress for QA Problem Set 2 Solutions

Useful examples can be found here: https://github.com/cypress-io/cypress-example-recipes

1 Session and State Management

Problem 1.1: Preserve login session across multiple tests

```
beforeEach(() => {
    cy.session('loginSession', () => {
        cy.visit('/login');
        cy.get('#email').type('qa@example.com');
        cy.get('#password').type('securepass');
        cy.get('#login-btn').click();
        cy.url().should('include', '/dashboard');
    });
});

it('should access protected page after login', () => {
    cy.visit('/settings');
    cy.get('h1').should('contain', 'Settings');
});
```

Problem 1.2: Store auth token and use in API test

```
let authToken;

before(() => {
    cy.request('POST', '/api/login', {
        email: 'qa@example.com',
```

```
        password: 'securepass'
    }).then((resp) => {
        authToken = resp.body.token;
    });
});

it('should fetch profile data with token', () => {
    cy.request({
        method: 'GET',
        url: '/api/profile',
        headers: {
            Authorization: `Bearer ${authToken}`
        }
    }).then((resp) => {
        expect(resp.status).to.eq(200);
        expect(resp.body).to.have.property('email');
    });
});
```

2 File Upload and Download

Problem 2.1: Upload a CSV file and validate response

```
import 'cypress-file-upload';

it('should upload a CSV file', () => {
    cy.visit('/upload');
    cy.get('input[type="file"]').attachFile('sample-data
       .csv');
    cy.get('#upload-btn').click();
    cy.get('.upload-success').should('contain', 'Upload
       complete');
});
```

Problem 2.2: Download a file and check filename

```
it('should trigger download with correct filename', ()
   => {
    cy.intercept('GET', '/files/report.pdf').as('
       fileDownload');
    cy.get('#download-btn').click();
    cy.wait('@fileDownload').its('response.headers')
        .should('have.property', 'content-disposition')
        .and('include', 'report.pdf');
});
```

3 Assertions and Conditional Logic

Problem 3.1: Conditionally test based on element existence

```
it('should click alert banner if visible', () => {
    cy.get('body').then($body => {
        if ($body.find('.alert-banner').length) {
            cy.get('.alert-banner').click();
            cy.get('.modal').should('be.visible');
        }
    });
});
```

Problem 3.2: Validate dynamic UI updates after API interaction

```
it('should reflect API change in UI', () => {
    cy.intercept('POST', '/api/toggle-feature', {
        statusCode: 200,
        body: { enabled: true }
    }).as('toggle');

    cy.get('#feature-toggle-btn').click();
    cy.wait('@toggle');
    cy.get('.feature-status').should('contain', 'Enabled
       ');
});
```

Problem 3.3: Test UI behavior based on local storage flag

```
it('should skip intro modal if dismissed before', () =>
   {
    cy.visit('/', {
        onBeforeLoad(win) {
            win.localStorage.setItem('hideIntroModal', '
               true');
        }
```

```
    });

    cy.get('#intro-modal').should('not.exist');
});
```

4 Modals, Tabs, and UI States

Problem 4.1: Ensure modal closes when clicking outside

```
it('should close the modal on outside click', () => {
    cy.get('#open-modal-btn').click();
    cy.get('#modal').should('be.visible');
    cy.get('body').click(0, 0); // Click outside modal
    cy.get('#modal').should('not.exist');
});
```

Problem 4.2: Switch tabs and validate content

```
it('should switch to the Reports tab', () => {
    cy.get('#tab-reports').click();
    cy.get('.tab-content').should('contain', 'Monthly
       Reports');
});
```

5 Pagination and Lazy Loading

Problem 5.1: Validate pagination loads more items

```
it('should load next page of results', () => {
    cy.get('.result-item').should('have.length', 10);
    cy.get('#next-page-btn').click();
    cy.get('.result-item').should('have.length.
       greaterThan', 10);
});
```

Problem 5.2: Infinite scroll triggers more content

```
it('should load more items on scroll', () => {
    cy.scrollTo('bottom');
    cy.wait(1000);
    cy.get('.infinite-item').its('length').should('be.gt
       ', 20);
});
```

6 APIs and GraphQL

Problem 6.1: Intercept GraphQL query and assert variables

```
it('should call GraphQL with correct variables', () => {
    cy.intercept('POST', '/graphql', (req) => {
        expect(req.body.operationName).to.eq('
           GetUserProfile');
        expect(req.body.variables.id).to.eq('user-123');
    }).as('graphqlRequest');

    cy.get('#fetch-profile-btn').click();
    cy.wait('@graphqlRequest');
});
```

Problem 6.2: Mock GraphQL mutation response

```
it('should mock mutation and verify UI', () => {
    cy.intercept('POST', '/graphql', (req) => {
        if (req.body.operationName === 'UpdateStatus') {
            req.reply({
                data: { updateStatus: { success: true }
                   }
            });
        }
    }).as('updateStatus');

    cy.get('#status-toggle').click();
    cy.wait('@updateStatus');
    cy.get('.status-msg').should('contain', 'Status
       updated');
```

```
});
```

7 Timing and Retry Strategies

Problem 7.1: Wait and assert on delayed element

```
it('should wait for delayed banner to appear', () => {
    cy.get('#delayed-banner', { timeout: 8000 }).should
       ('be.visible');
});
```

Problem 7.2: Retry logic for flaky backend error

```
it('should retry API on failure and eventually succeed',
    () => {
    let attempt = 0;

    cy.intercept('GET', '/api/retry-endpoint', (req) =>
       {
        attempt++;
        if (attempt < 3) {
            req.reply({ statusCode: 500 });
        } else {
            req.reply({ statusCode: 200, body: { status:
                'ok' } });
        }
    }).as('retryAPI');

    cy.visit('/retry-page');
    cy.wait('@retryAPI');
    cy.get('.status-indicator').should('contain', 'ok');
});
```