

# Code Summariser and Evaluator

520 final project

## Testing document

Submitted by :

Final project group 10

### Team Members:

- Ashwini Ramesh Kumar - [ashwinirames@umass.edu](mailto:ashwinirames@umass.edu) - 34025576
- Aarthi Nunna - [anunna@umass.edu](mailto:anunna@umass.edu) - 34042804
- Dhamini - [ddevaraj@umass.edu](mailto:ddevaraj@umass.edu) - 34048503
- Janani - [jpasham@umass.edu](mailto:jpasham@umass.edu) - 34023678
- Rashmi Vagha - [rvagha@umass.edu](mailto:rvagha@umass.edu) - 33617396

### Tests done:

1. Unit testing of components using the react library Jest.
2. Integration testing of API calls using Postman.
3. Equivalence Testing/ Blackbox testing

- **Unit Testing:**

1. **Component : AccountSettings.jsx**

#### **Test 1 :**

To check if it renders the account information correctly.

#### **Expected output:**

The rendered page must have “Account Information”, “Admin Name”  
,”Admin Password” ,” Change Password” Labels.

#### **Output Obtained:**

All labels are obtained correctly. **Test passed.**

#### **Test 2:**

To check if the UI shows change password form when the change password button is clicked.

#### **Expected output:**

The page renders the form.

#### **Output Obtained:**

Form is rendered correctly. **Test passed.**

#### Test Code:

```
AccountSettings.test.jsx / ...
import React from 'react';
import { render, fireEvent } from '@testing-library/react';
import AccountSettings from './AccountSettings.jsx';

describe('AccountSettings component', () => {
  test('renders account information correctly', () => {
    const { getByLabelText, getByText } = render(<AccountSettings />);

    expect(getByLabelText('Account Information')).toBeInTheDocument();
    expect(getByLabelText('Admin Name:')).toBeInTheDocument();
    expect(getByLabelText('Admin Password:')).toBeInTheDocument();
    expect(getByText('Change Password')).toBeInTheDocument();
  });

  test('shows change password form when Change Password button is clicked', () => {
    const { getByText, getByLabelText } = render(<AccountSettings />);

    fireEvent.click(getByText('Change Password'));

    expect(getByLabelText('New Password:')).toBeInTheDocument();
    expect(getByText('Submit New Password')).toBeInTheDocument();
  });
});
```

## 2. Component: AddAdmin

**Test 1:** Renders form fields -Add Admin, Admin Name, Password, Add Administrator correctly

**Expected output:** Renders all fields and labels correctly.

**Output obtained: Test passed.**

**Test 2:** Updates the usestate hook variables when user input is given.

**Expected Output:** Updates all variables in the UI correctly.

**Output Obtained: Test passed.**

#### Test Code:

```

import React from 'react';
import { render, fireEvent } from '@testing-library/react';
import AddAdmin from './AddAdmin';

describe('AddAdmin component', () => {
  test('renders form fields correctly', () => {
    const { getByLabelText, getByText } = render(<AddAdmin />);

    expect(getByText('Add Admin')).toBeInTheDocument();
    expect(getByLabelText('Name')).toBeInTheDocument();
    expect(getByLabelText('Password')).toBeInTheDocument();
    expect(getByText('Add Administrator')).toBeInTheDocument();
  });

  test('updates input values on user input', () => {
    const { getByLabelText } = render(<AddAdmin />);

    const nameInput = getByLabelText('Name');
    const passwordInput = getByLabelText('Password');

    fireEvent.change(nameInput, { target: { value: 'admin' } });
    fireEvent.change(passwordInput, { target: { value: 'password' } });

    expect(nameInput.value).toBe('admin');
    expect(passwordInput.value).toBe('password');
  });
});

```

### 3. Component: AdminDashboard

**Test1** :Renders navigation bar correctly.

**Expected Output:** The component must render the navigation bar to the left of the UI correctly with all labels and divisions.

**Output obtained: Test passed.**

**Test Code:**

```
import React from 'react';
import { render, fireEvent } from '@testing-library/react';
import AdminDashboard from './AdminDashboard';

describe('AdminDashboard component', () => {
  test('renders navigation bar correctly', () => {
    const { getByText } = render(<AdminDashboard />);

    expect(getByText('Admin Mode')).toBeInTheDocument();
    expect(getByText('Add an Admin')).toBeInTheDocument();
    expect(getByText('View User Stats')).toBeInTheDocument();
    expect(getByText('Change to user mode')).toBeInTheDocument();
    expect(getByText('Account Settings')).toBeInTheDocument();
  });
});
```

#### Cumulative Test results (running npm test)

```
PASS src/AccountSettings.test.jsx
PASS src/AddAdmin.test.jsx
PASS src/AdminDashboard.test.js

Test Suites: 3 passed, 3 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        6.613 s
Ran all test suites.
```

- **Integration Testing:**

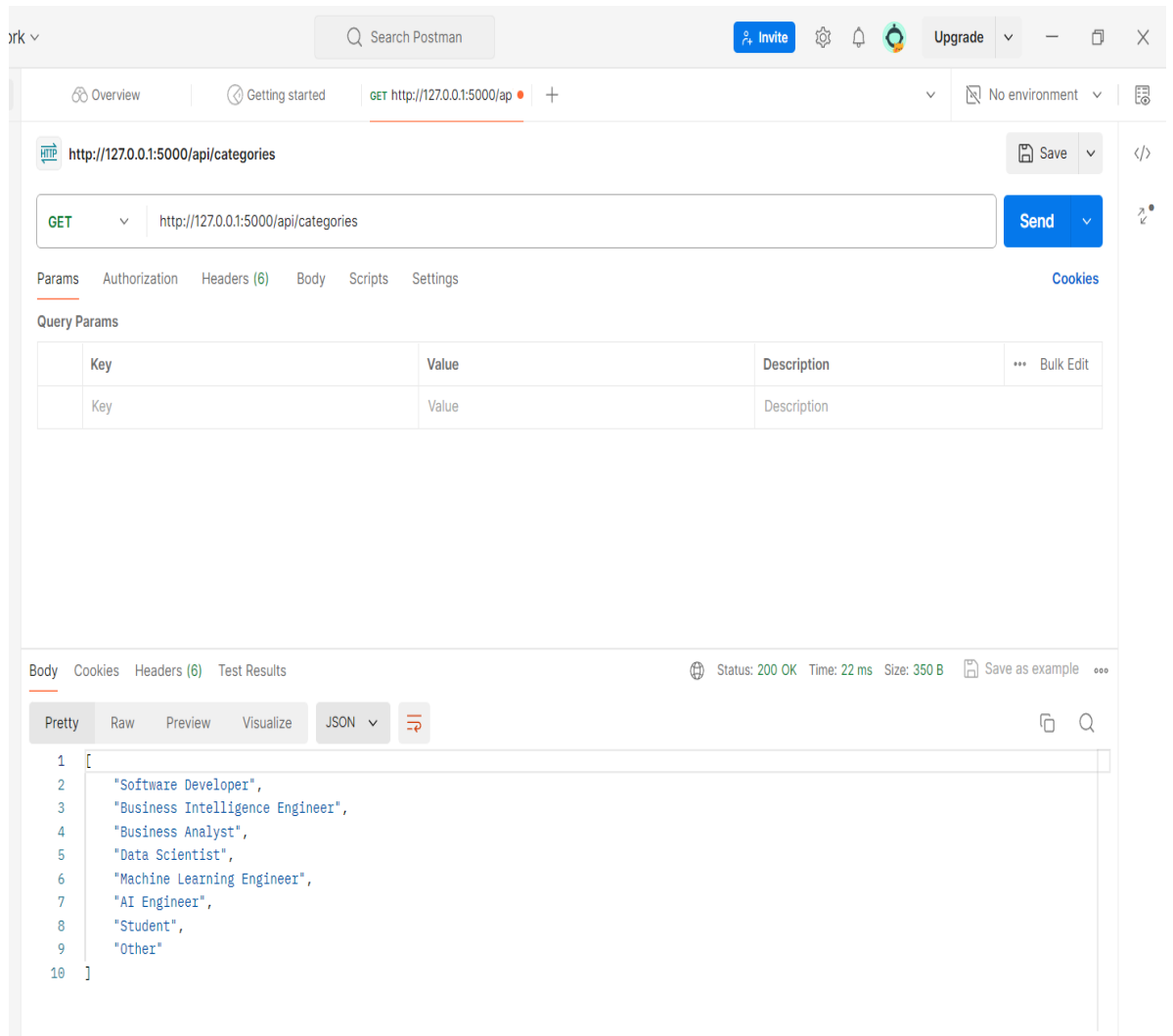
We are using PostMan to test different API's which perform different functionalities that connect the frontend to the backend of our application.

1. **Test Case 1:** Renders Categories

**Request Used in PostMan :** <http://127.0.0.1:5000/api/categories> - GET  
(Request Type)

**Expected :** List of Categories in JSON

**Output Obtained:** Status : 200( Test Case Passed)



2. **Test Case 2:**Retrieving the values which user has submitted for feedback  
**Request Used in PostMan:** <http://127.0.0.1:5000/api/singleUserEval> -  
GET(Request Type)  
**Output Expected :** JSON with the feedback values

## Output Obtained: Status : 200 ( Test Case Passed)

The screenshot shows the Postman interface for a GET request to `http://127.0.0.1:5000/api/singleUserEval`. The request is successful, returning a status of 200 OK, a response time of 13 ms, and a size of 256 B. The response body is displayed in JSON format:

```
{
  "consistency": 0,
  "count": 0,
  "naturalness": 0,
  "usefulness": 0
}
```

- **Equivalence Testing (Blackbox testing) :**

**Case 1 - First Name and/or Last Name missing during a new user Sign Up**

**Expected Output** - System should display an error message stating both these fields are mandatory

**Actual Output** - Error message displayed correctly. **Test Passed.**

**Screenshot -**

The screenshot shows a sign-up form with the following fields and error messages:

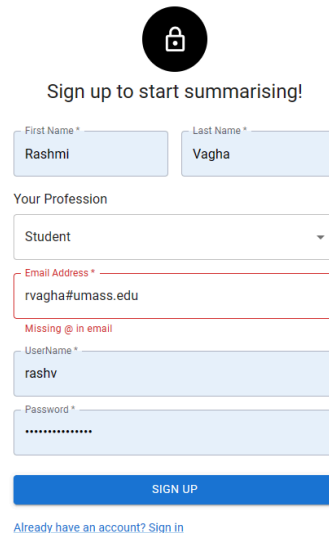
- First Name \***: Error message "First Name is required" (in red).
- Last Name \***: Error message "Last Name is required" (in red).
- Your Profession**: A dropdown menu with "Student" selected.
- Email Address \***: Text input field containing "rvagha@umass.edu".
- UserName \***: Text input field containing "rashv".
- Password \***: Password input field with masked characters "\*\*\*\*\*".
- SIGN UP**: A blue button at the bottom.
- Already have an account? Sign In**: A link at the bottom.

### Case 2 - Email Address invalid during a new user Sign Up

**Expected Output** - System should display an error message stating that a proper format of email address is expected.

**Actual Output** - Error message displayed stating @ is required in the email address. **Test Passed.**

**Screenshot -**



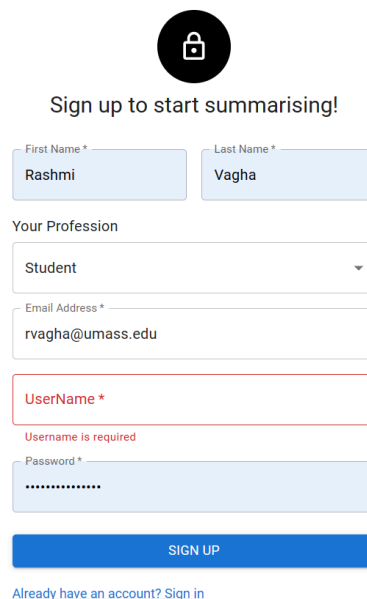
The screenshot shows a sign-up form titled "Sign up to start summarising!". The form includes fields for First Name (Rashmi), Last Name (Vagha), Your Profession (Student), Email Address (rvagha#umass.edu), Username (rashv), and Password (masked with dots). A red error message "Missing @ in email" is displayed below the Email Address field. A blue "SIGN UP" button is at the bottom, and a link "Already have an account? Sign In" is below it.

### Case 3 - Username not entered during a new user Sign Up

**Expected Output** - System should display an error message stating that username is a mandatory field.

**Actual Output** - Correct error message displayed. **Test Passed.**

**Screenshot -**




The screenshot shows the same sign-up form as Case 2, but with the Username field (rashv) empty. A red error message "Username is required" is displayed below the Username field. The Email Address field now contains "rvagha@umass.edu". The "SIGN UP" button and "Sign In" link are also present.

**Case 4** - Password entered during a new user Sign Up is too short or does not follow the password requirements.

**Expected Output** - System should display an error message stating the password requirements and that currently entered password does not match them.

**Actual Output** - Correct error message displayed. **Test Passed.**

**Screenshot -**



Sign up to start summarising!

First Name \*  
Rashmi

Last Name \*  
Vagha

Your Profession  
Student

Email Address \*  
rvagha@umass.edu

UserName \*  
rashv

Password \*  
\*\*\*\*\*

Password must be at least 8 characters long and contain at least one uppercase letter, one lowercase letter, and one special character

SIGN UP

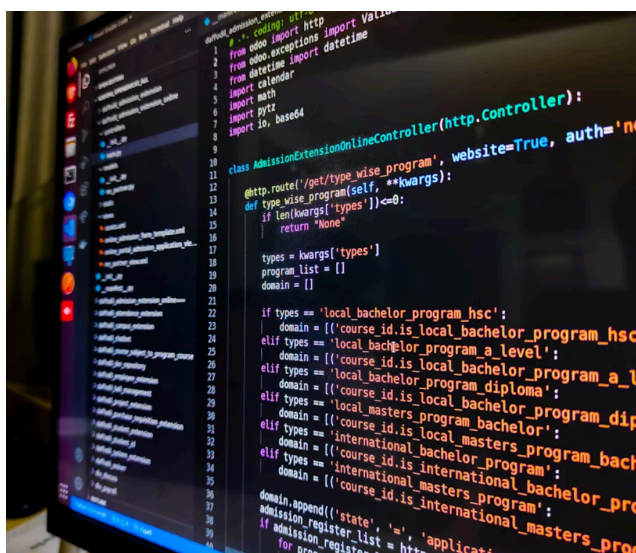
[Already have an account? Sign in](#)

**Case 5** - Trying to login with a username that does not exist on Sign In page


**Expected Output** - System should display an error message stating the account does not exist and any new user should sign up.

**Actual Output** - Correct error message displayed. **Test Passed.**

**Screenshot -**



```
1 from odoo import http
2 from odoo.exceptions import ValidationError
3 from datetime import datetime
4 import math
5 import pytz
6 import io, base64
7
8 class AdmissionExtensionOnlineController(http.Controller):
9
10     @http.route('/get/type_wise_program', website=True, auth='no')
11     def type_wise_program(self, **kwargs):
12         if len(kwargs['types']) <= 0:
13             return "None"
14
15         types = kwargs['types']
16         program_list = []
17         domain = []
18
19         if types == 'local_bachelor_program_hsc':
20             domain = [('course_id.is_local_bachelor_program_hsc', True)]
21         elif types == 'local_bachelor_program_a_level':
22             domain = [('course_id.is_local_bachelor_program_a_level', True)]
23         elif types == 'local_bachelor_program_diploma':
24             domain = [('course_id.is_local_bachelor_program_diploma', True)]
25         elif types == 'local_bachelor_program_dip':
26             domain = [('course_id.is_local_bachelor_program_dip', True)]
27         elif types == 'international_bachelor_program_bach':
28             domain = [('course_id.is_international_bachelor_program_bach', True)]
29         elif types == 'international_bachelor_program_pro':
30             domain = [('course_id.is_international_bachelor_program_pro', True)]
31         domain.append(('state', '=', 'application'))
32         admission_register_list = http.request._request.session.get('admission_register_list', [])
33         for program in program_list:
```



Sign in to start summarising!

Username \*  
rashv

Password \*  
\*\*\*\*\*

SIGN IN

Account does not exist, sign up now to start summarising!  
[Forgot password?](#) [Don't have an account? Sign Up](#)

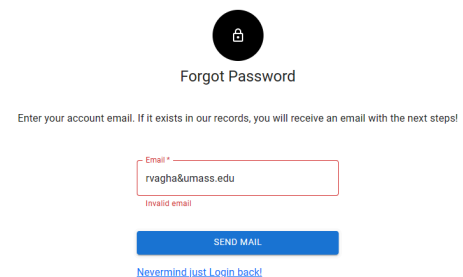


**Case 6** - Email address entered during Forgot Password page is invalid.

**Expected Output** - System should display an error message stating that a proper format of email address is expected.

**Actual Output** - Correct error message displayed. **Test Passed.**

**Screenshot -**

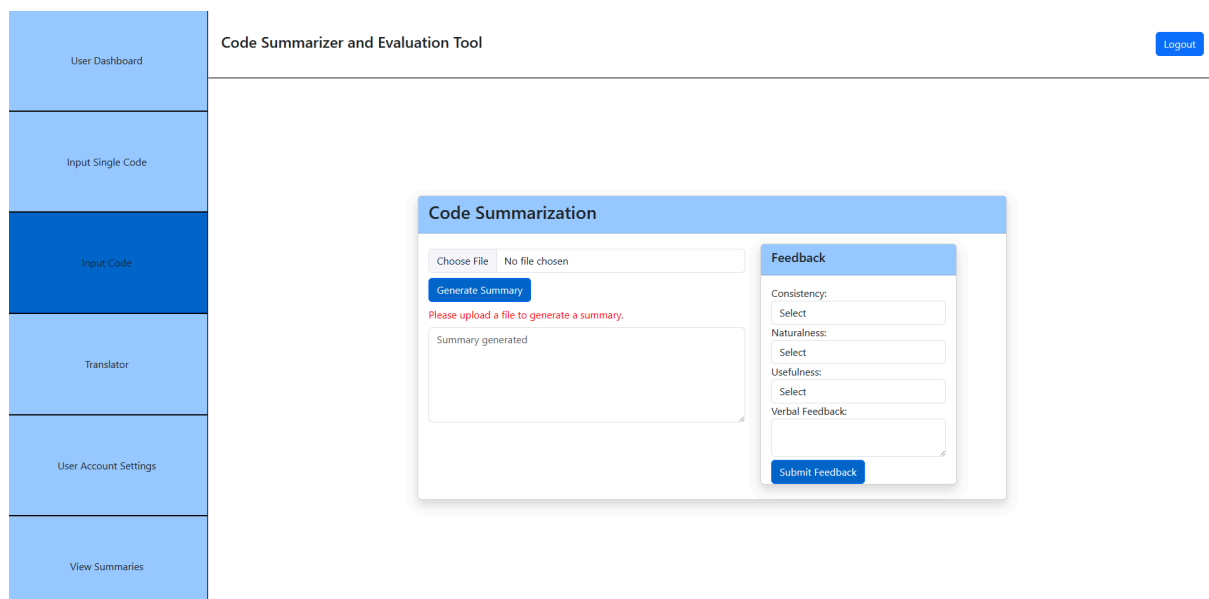


**Case 7** - Trying to generate a code summary without uploading the source code on the Code Summarization page.

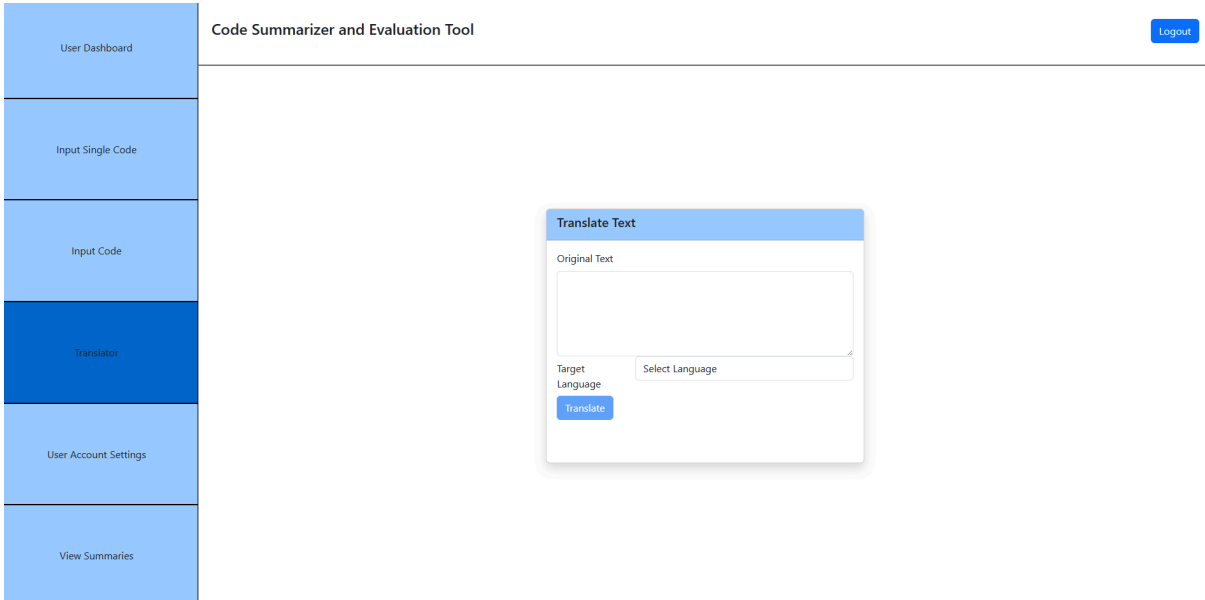
**Expected Output** - System should display an error message asking the user to upload an input file to generate a summary.

**Actual Output** - Error message displayed requesting the user to upload a file. **Test Passed.**

**Screenshot -**



**Case 8** - Trying to translate without entering the summary on the Translate Text page.  
**Expected Output** - The translate button should not be enabled until the summary in the source language is entered.  
**Actual Output** - The Translate button is displayed. **Test Passed.**  
**Screenshot -**



**Case 9** - Translating the summary on the Translate Text page.  
**Expected Output** - The translate button should be enabled after entering the summary in source language and choosing the destination language and the correct summary should be generated after clicking 'Translate'.  
**Actual Output** - The Translate button is enabled and correct summary is generated. **Test Passed.**  
**Screenshot**

