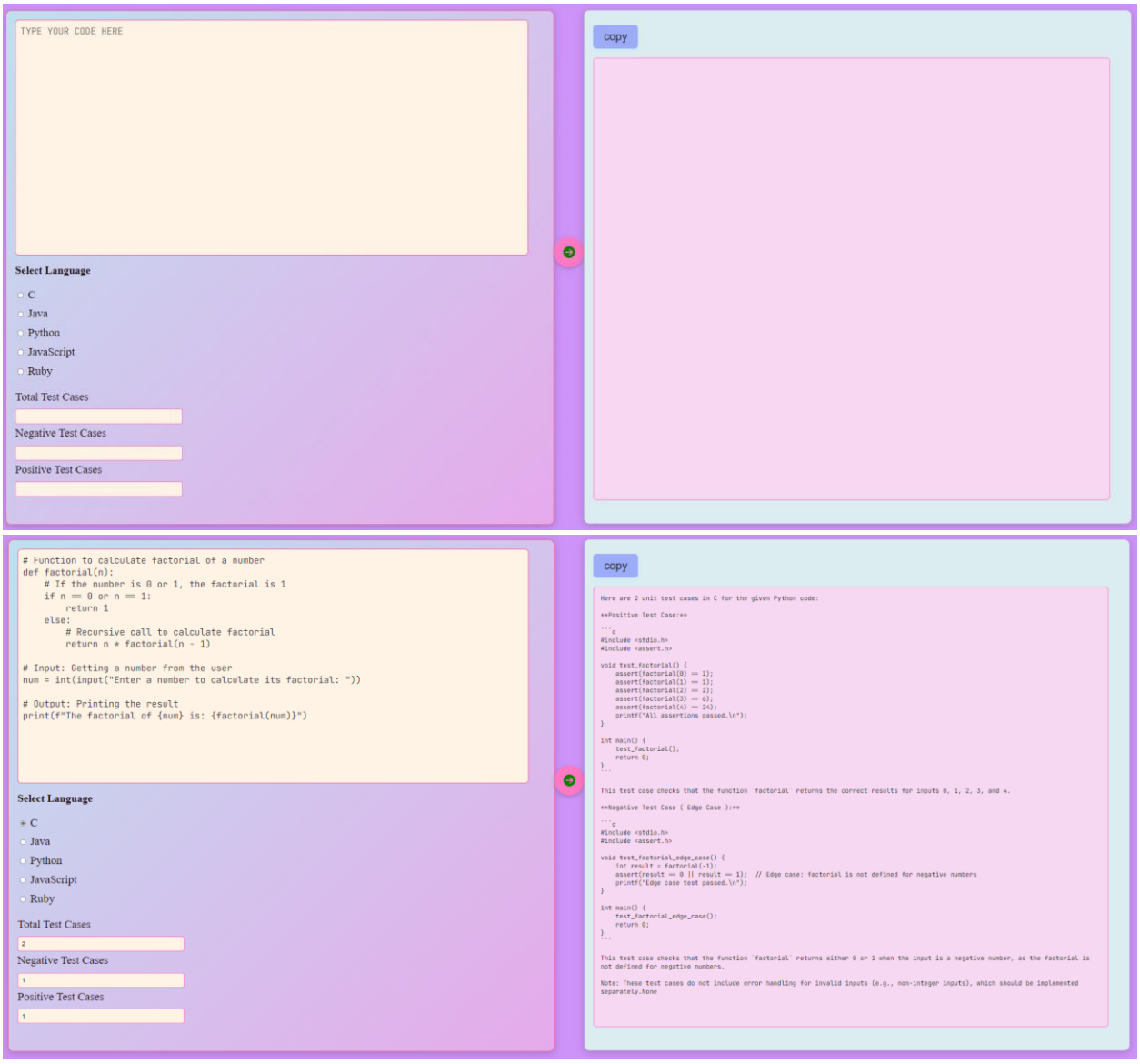
Design and Approach Document for Unit Test Case Generation

Objective: The application automates the generation of unit test cases based on user-provided code. It allows users to input code, select a programming language, and specify test case requirements (total, positive, negative) to generate corresponding unit test cases.

Architecture Overview:



1. **Frontend (UI):**
    a. **Technology:** React.js, HTML, CSS.
    b. Components:

        i. **InputComponent:** Takes user input for code, language, and test case specifications (total, positive, negative).
        ii. **OutputComponent:** Displays generated test cases or error messages and allows copying the result.
    c. **Key Features:**
        i. Code input field, language selection, and test case input fields.
        ii. Real-time validation and error messages.
2. **Backend (API):**
    a. **Technology:** Flask/Django (Python).
    b. **Functions:** Validates code, generates unit test cases based on user input using AI-powered templates, and returns the result.
    c. **Endpoints:** API to validate code and generate test cases.

Tech Stack:

- **Frontend:** React.js, HTML, CSS, FontAwesome for icons.
- **Backend:** Flask/Django (Python), AI integration for code validation and test case generation.
- **Libraries:** Fetch API, FontAwesome icons for UI elements.

Key Features:

1. **Input Validation:** Ensures valid inputs (code, language, and test cases), including checks for negative values and mismatched sums for positive and negative cases.
2. **Test Case Generation:** Generates unit test cases in specified programming language (Java, Python, JavaScript, etc.).
3. **Real-Time Feedback:** Provides error messages for invalid inputs and success/failure states.
4. **Copy to Clipboard:** Users can copy generated test cases with a single click.

Component Breakdown:

1. **InputComponent:**
    a. Manages user input, validates inputs, and tracks errors.
    b. Submits data to backend for test case generation.
2. **OutputComponent:**
    a. Displays generated test cases or error messages.
    b. Allows users to copy results to clipboard.

Conclusion: The application automates the creation of unit test cases, making it easier for developers to generate tests for their code across multiple languages. It provides real-time input validation, generates tests using AI, and includes user-friendly features like error handling and copy-to-clipboard functionality.