Summary Week 11

By Hiren Rupchandani

The professor started the lecture by talking about the importance of effective software testing in software development. While testing is crucial, it can be challenging due to various factors. For example, developers need to identify as many bugs as possible while minimizing resource use, which requires prioritizing tests and using different testing techniques. Also, conducting exhaustive testing is impractical, and bugs tend to be distributed unevenly across modules, which means that some modules require more rigorous testing than others. Additionally, the context of the software being tested, such as whether it's a mobile app or software used in rockets, affects how test cases are created.

There are different types of testing that developers use. Unit testing focuses on testing individual units of a system in isolation, which is quick and easy to control, but may not accurately reflect the system's behavior due to interactions between different classes. Integration testing, on the other hand, involves testing multiple components of a system together and emphasizing their interactions. System testing tests the entire system, but it can be slow, difficult to set up, and prone to errors.

To decide which testing approach to take, developers use models such as the testing pyramid and the testing trophy. The testing pyramid suggests testing the most at the base, which is unit tests, followed by integration tests, and fewer system tests at the top. The testing trophy emphasizes the value of static checks such as linters and type checkers and acknowledges the importance of unit and integration testing.

Different companies have different testing approaches depending on their context and requirements. For instance, Google prioritizes unit testing, while Spotify prefers integration testing. To generate tests, developers need to understand the requirements and use specification-based testing techniques to explore the input variables and their interactions systematically. The seven-step approach for specification testing involves understanding requirements, analyzing inputs and outputs, identifying partitions and boundaries, devising concrete test cases, implementing them as automated tests, and augmenting the test suite with creativity and experience. Boundary identification is the most challenging part of testing, as bugs often hide there. Even simple programs require careful selection of what to test due to the large number of test cases.

This lecture was followed by a closed-book quiz during the lecture timings.

In Thursday's lecture, the professor talked about client-side frameworks in web development, which provide tooling to improve developer experience and allow for testing and linting. Components are used to abstract different parts of user interfaces. Modern web applications use single page apps and client-side routing to continually update the DOM, while complex SPAs require routing functionality to manage unique views. Developers use a JavaScript package called

ReactJS to create user interfaces. It offers a component-based approach, where each component represents a part of the user interface that can be used elsewhere in the program. By changing the UI in response to changes in the data, React enables developers to build dynamic and interactive web apps. Here's an example of a React component:

```
import React from 'react';

class Button extends React.Component {
  render() {
    return (
        <div>
            <button onClick={this.props.onClick}>
              {this.props.title }
            </button>
        </div>
    );
  }
}

export default Button;
```

References:

1. Lecture Slides
2. https://www.indeed.com/career-advice/career-development/testing-methodologies
3. https://react.dev/blog/2023/03/16/introducing-react-dev