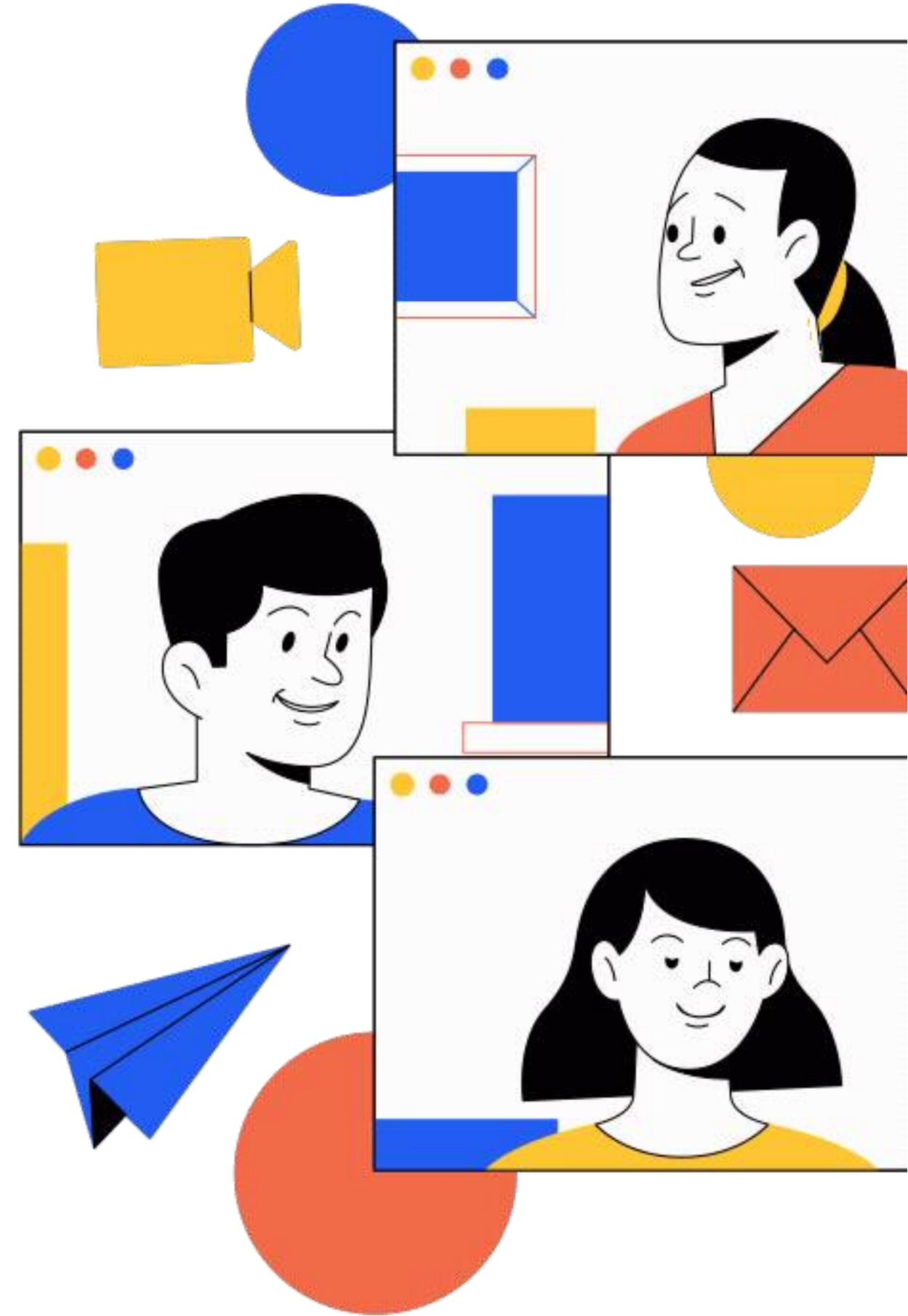


Week 9 - Employee Poll

One last ride ?

Ahmed Fouad Lotfy

React Session Lead



Agenda

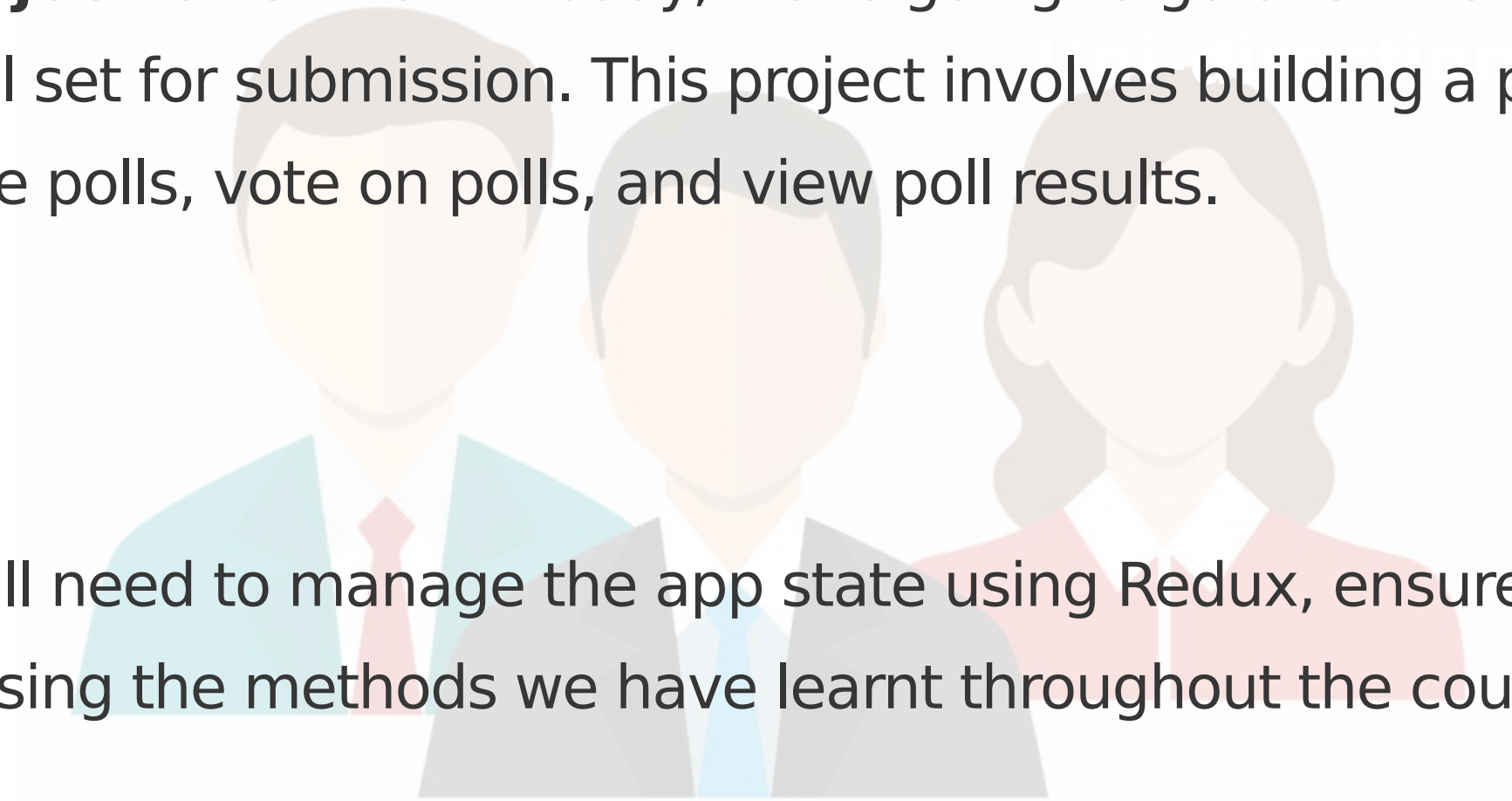


What we'll cover in this session

- Project Overview
- Breakdown of Requirements
- Application Functionality
- Architecture and State Management
- Testing Requirements

Project Overview

- **Employee Polls Project Overview:** Today, we're going to go over the Employee Polls project to make sure you're all set for submission. This project involves building a polling app where users can log in, create polls, vote on polls, and view poll results.
- **Requirements:** You'll need to manage the app state using Redux, ensure navigation works, and write unit tests. Using the methods we have learnt throughout the course.



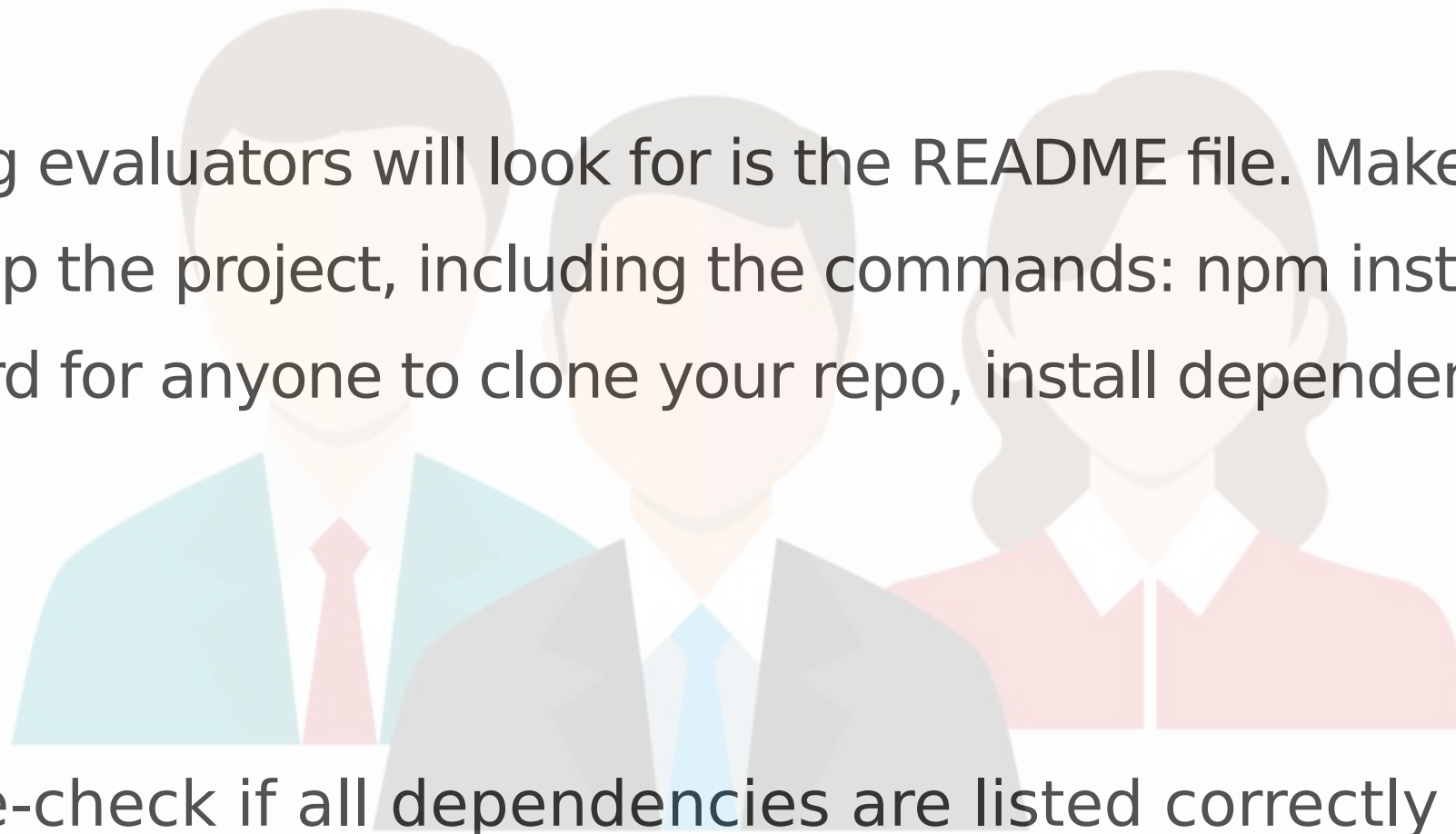
Breakdown of Requirements

- **Application setup**
- **Login Flow**
- **Home Page**
- **Poll Details**
- **Leaderboard**



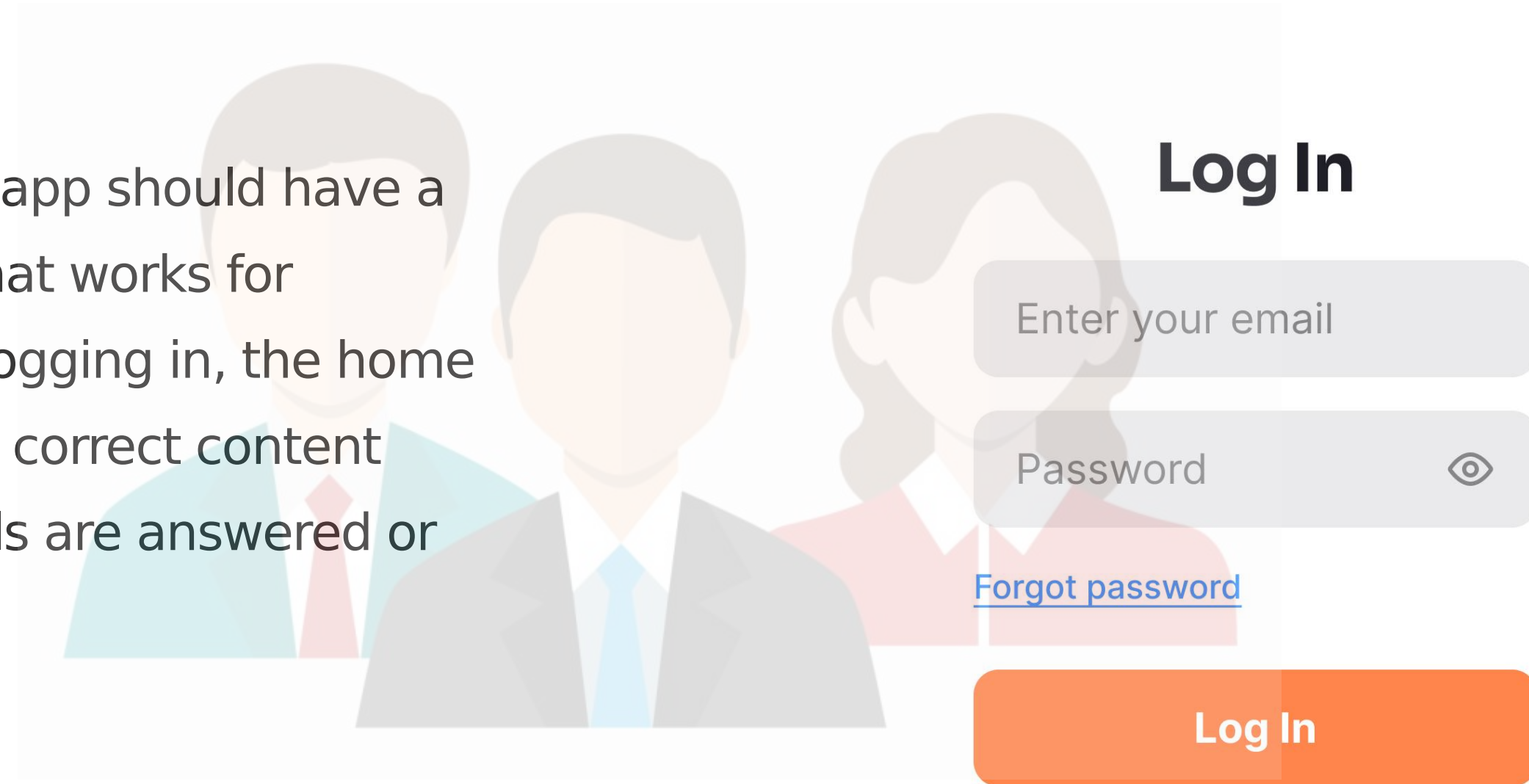
Application setup

- **Readme:** The first thing evaluators will look for is the README file. Make sure it has clear instructions for setting up the project, including the commands: `npm install` and `npm start`. It should be straightforward for anyone to clone your repo, install dependencies, and start the app without issues.
- **Package.json:** Double-check if all dependencies are listed correctly in `package.json`.



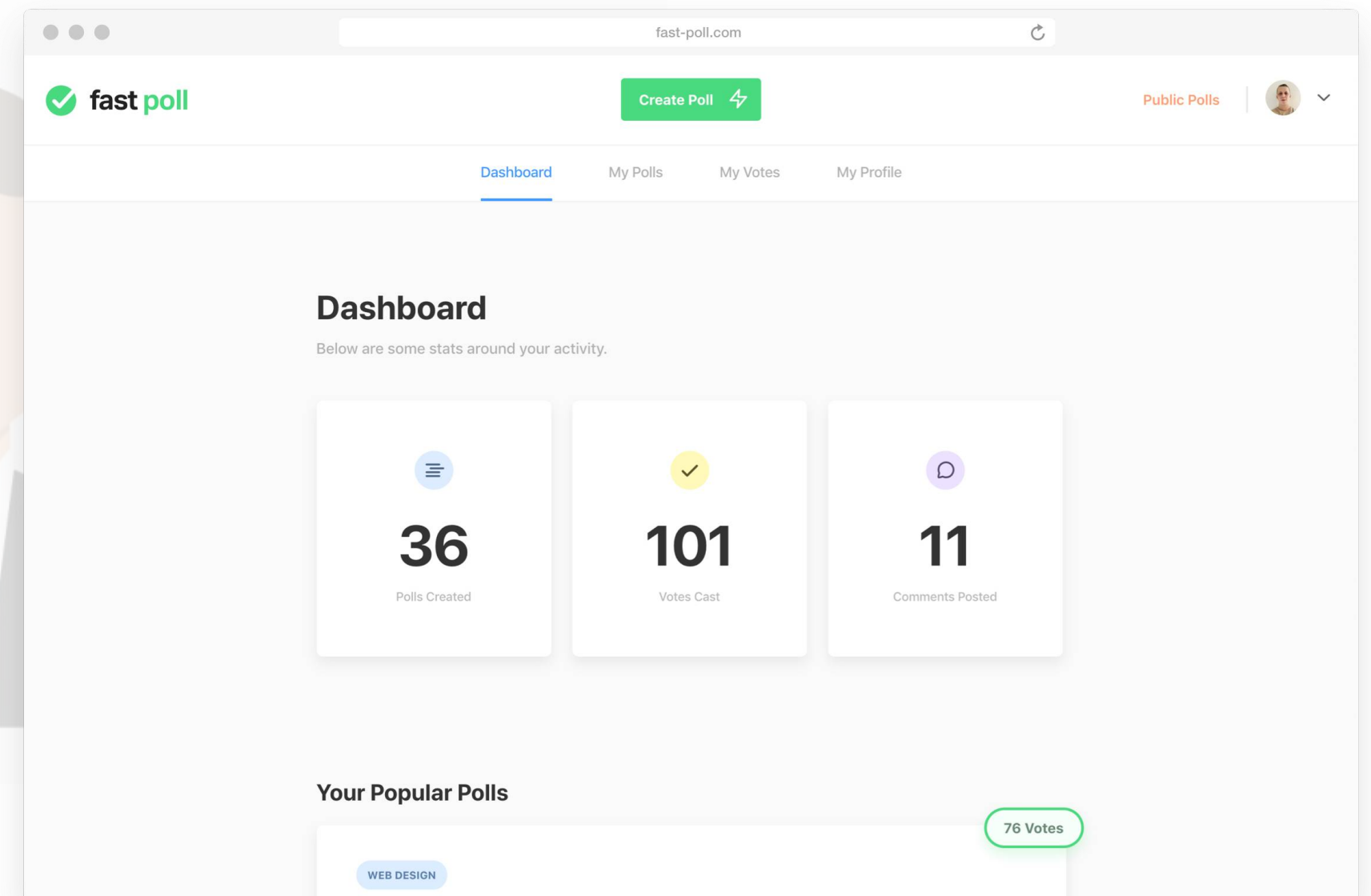
Login Flow

- **Login/Logout** : The app should have a login/logout system that works for different users. After logging in, the home page should show the correct content based on whether polls are answered or unanswered.



Home Page

- **Dashboard** : The home page should display two categories: Unanswered Polls and Answered Polls. Polls must be sorted by the timestamp (most recent first). Users should be able to navigate easily between the Polls, the Leaderboard, and Create New Poll.



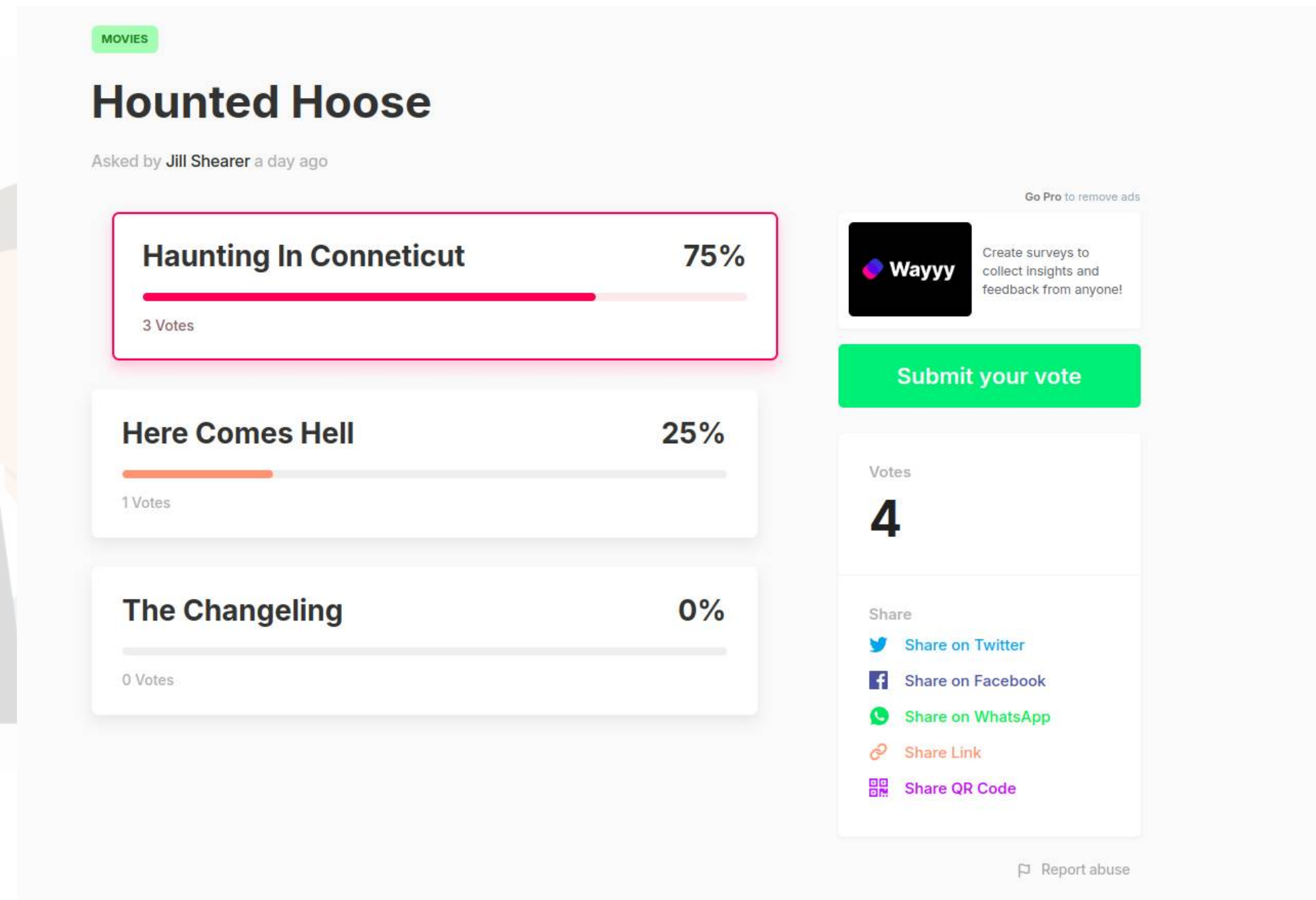
Check if your routing works properly to switch between views, and make sure the correct category is displayed for each poll.

<https://fast-poll.com/public/>

Poll Details

- **Poll :** When a user clicks on a poll, they should be able to see the full details of that poll. This includes the question text, avatar of the poll creator, and voting options. Once the user votes, the app should show the results: a percentage of how many people voted for each option, along with their avatar.

Include error handling for polls that don't exist — for instance, displaying a 404 page.

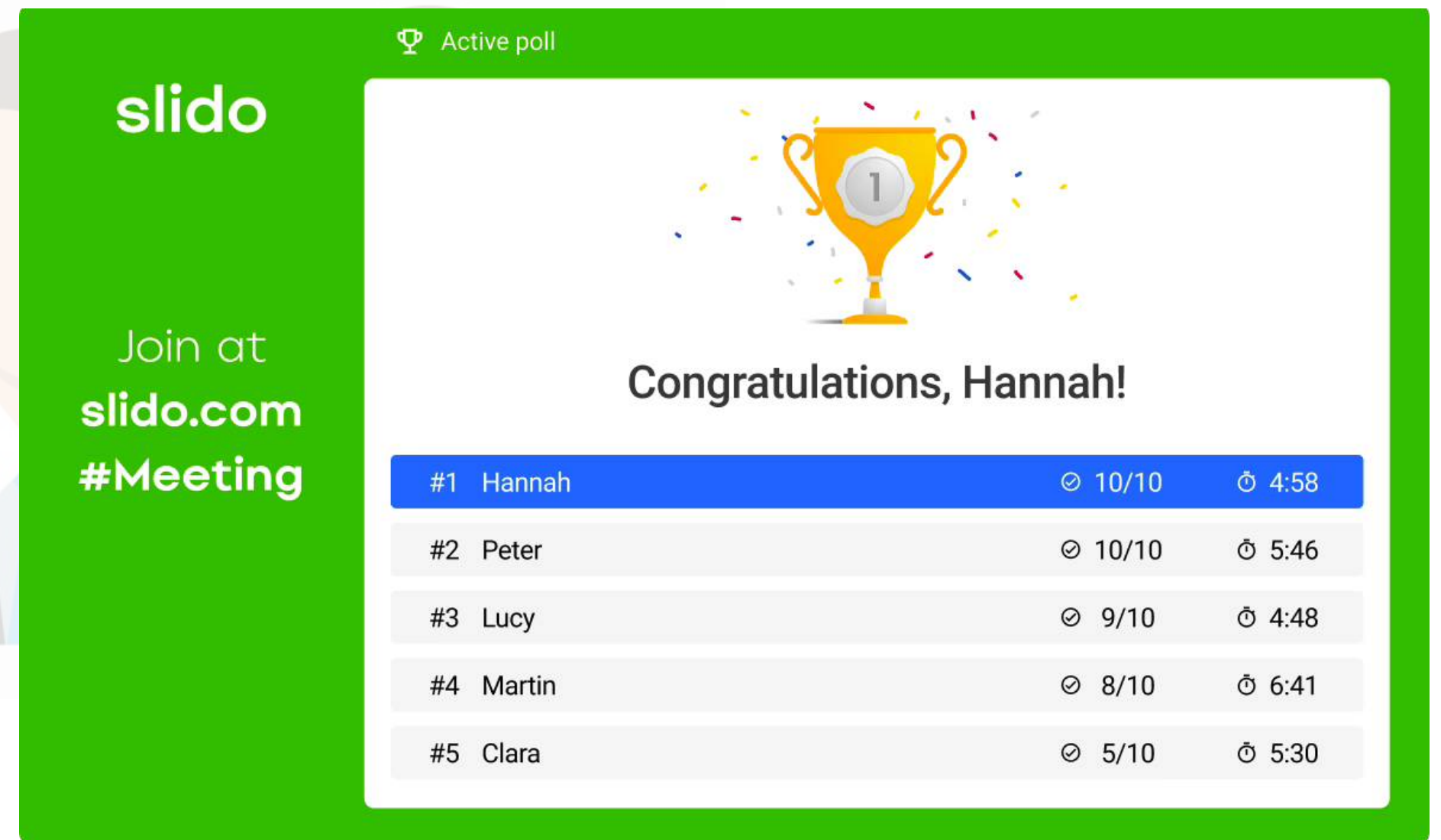


<https://fast-poll.com/poll/results/7d5edcdd>

Leaderboard

- **Ranking** : The Leaderboard should rank users based on two things: the number of questions they have asked and the number of questions they have answered. The users should be ranked in descending order, so the user with the highest activity is at the top. This data should be dynamically updated as users interact with the app.

Check it by testing it out by creating polls and voting with different users to see if the leaderboard updates correctly.

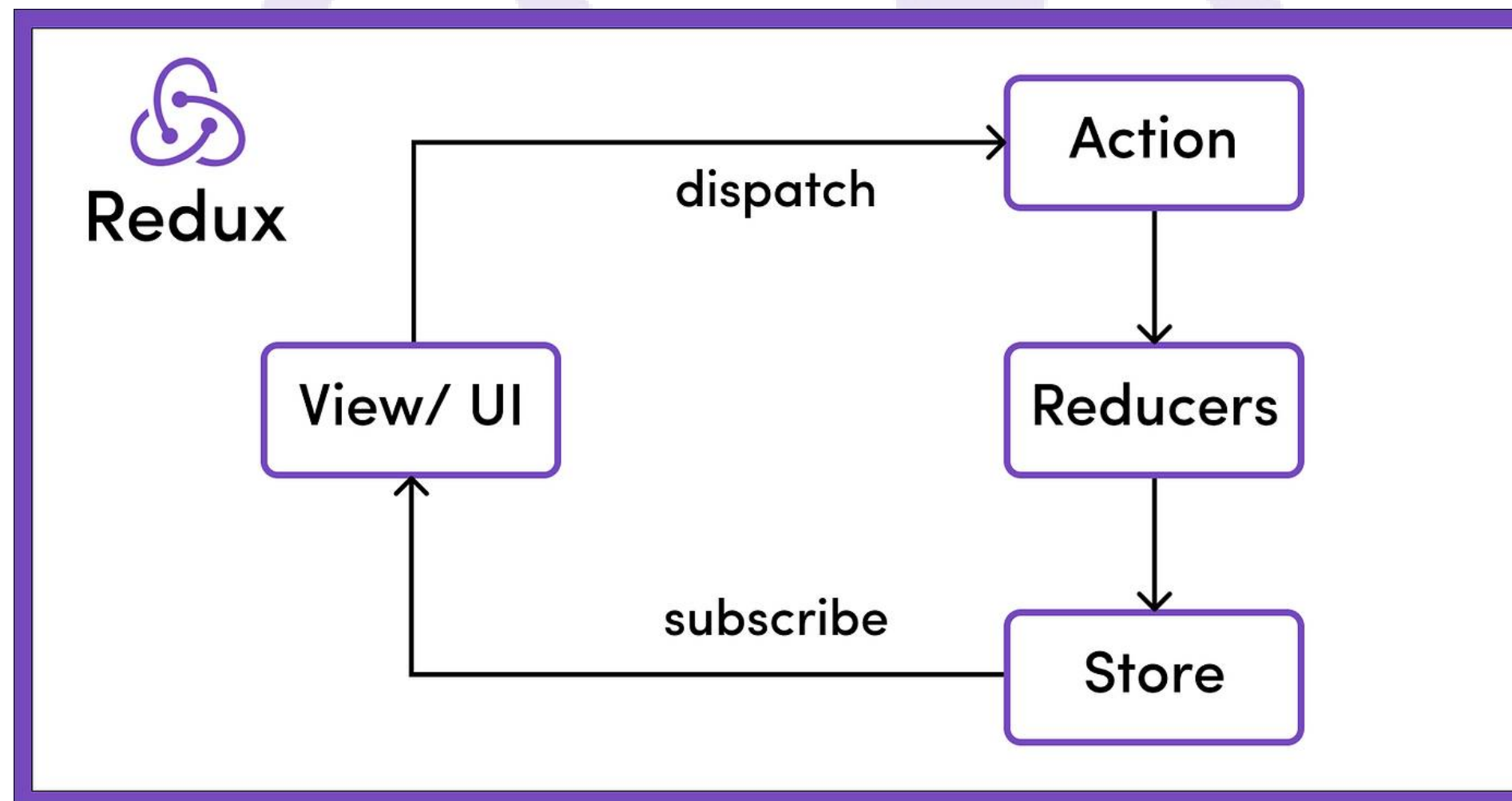


The screenshot shows the Slido app interface during an active poll. On the left, a green sidebar contains the 'slido' logo, the text 'Join at slido.com', and the hashtag '#Meeting'. The main area has a green header with 'Active poll' and a trophy icon. Below this, a large yellow trophy with the number '1' is displayed with the text 'Congratulations, Hannah!'. A leaderboard table follows, listing the top five participants with their names, scores, and times.

Rank	Name	Score	Time
#1	Hannah	10/10	4:58
#2	Peter	10/10	5:46
#3	Lucy	9/10	4:48
#4	Martin	8/10	6:41
#5	Clara	5/10	5:30

Redux Management

- **Managing State with Redux:** Your app's state should be managed entirely in the Redux store. Make sure your action creators and reducers are properly set up to handle poll creation, voting, and user authentication. Avoid directly mutating the state — always return a new state object.



Also, remember to separate your concerns: keep your components focused on rendering and delegate data logic to Redux.

Jest Testing

- **Testing:** For this project, you need to have at least 10 unit tests using Jest. Two of these tests should cover the asynchronous functions: `_saveQuestion()` and `_saveQuestionAnswer()`
- **Snapshot:** Use Jest to create snapshot tests for your components, and write DOM tests for interacting with UI elements.

Jest

Submission Checklist



Before you submit your project, run through this checklist:

1. Does your README file have clear installation and setup instructions?
2. Does the login/logout flow work properly?
3. Can you navigate between all views (home, leaderboard, poll details, etc.) without issues?
4. Are all required features implemented and tested (poll creation, voting, leaderboard, etc.)?
5. Do you have 10 unit tests with Jest, covering key functionality like saving questions and voting?

"If you can answer 'yes' to all of these, you're ready to submit!"

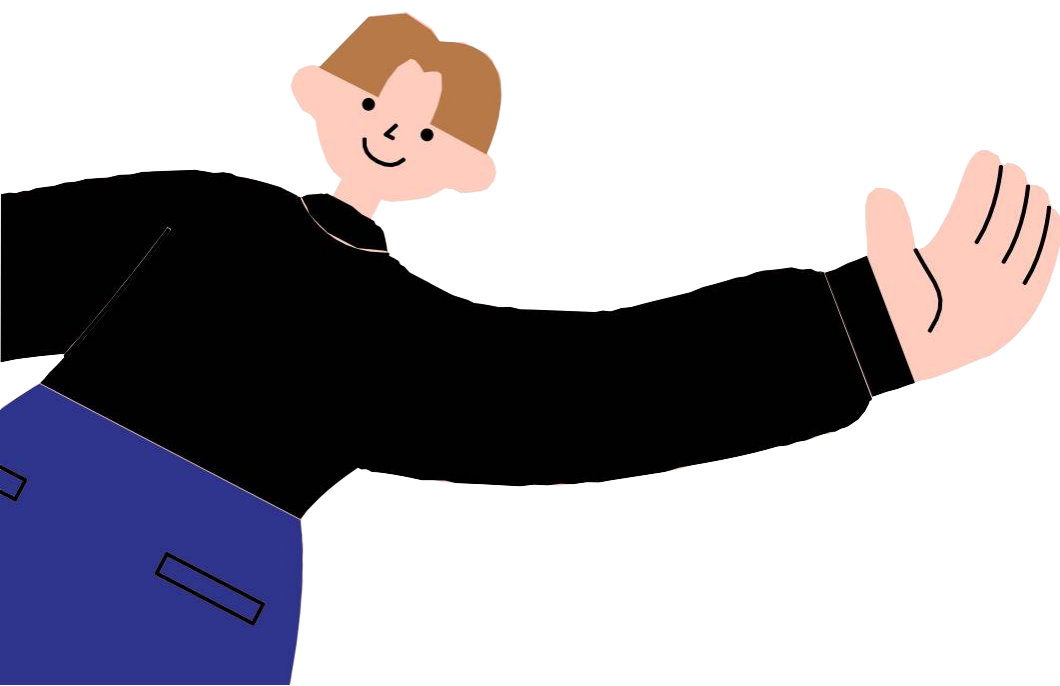
No Demo Time

For Today



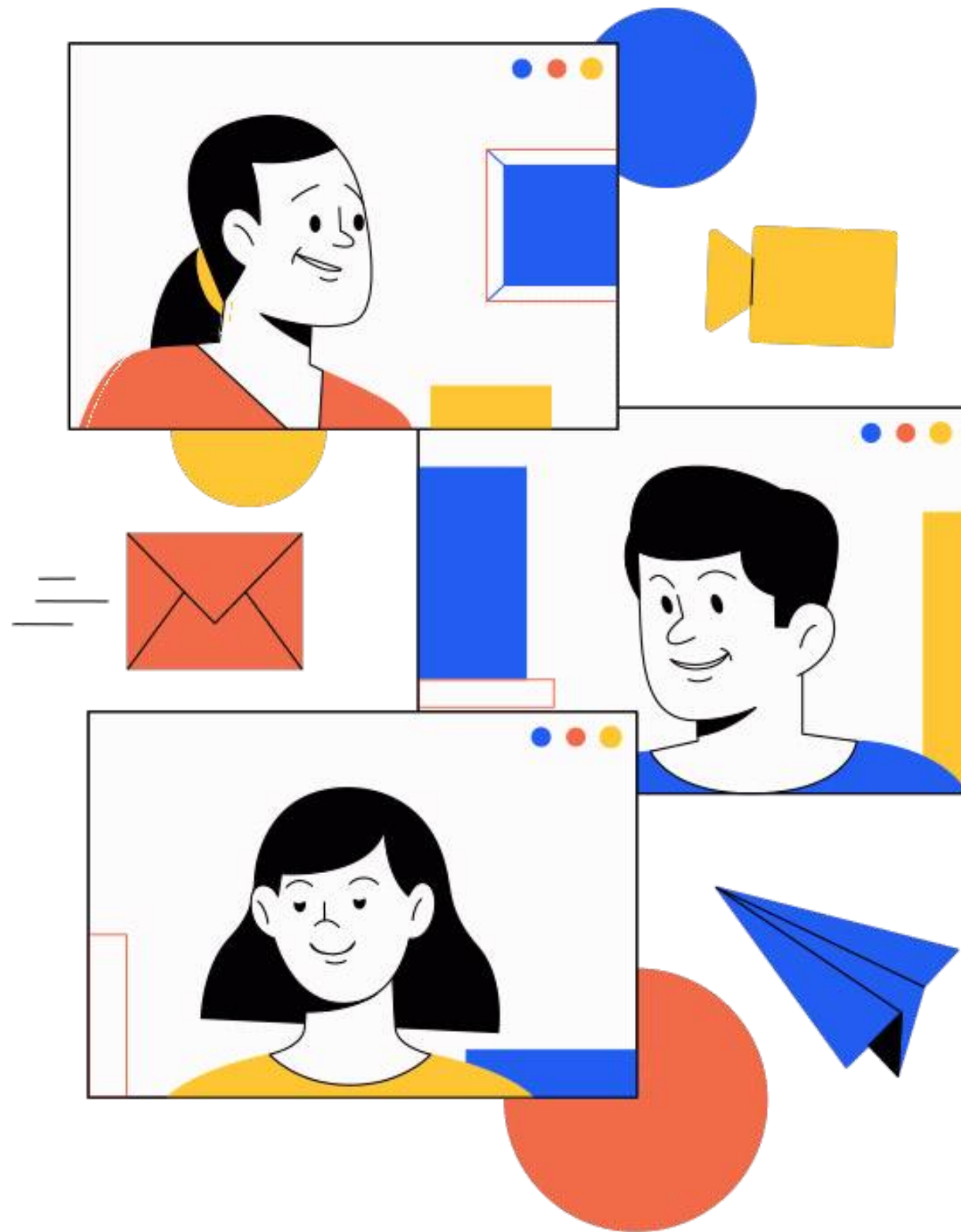


Any Questions?



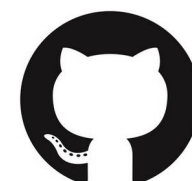
Next Session: React Native





Thank you for attending!

Feel free to email at a.lotfy@fci-cu.edu.eg or reach me at circle anytime for any questions or clarifications!



Follow me on Github [@ahmeddx Fouad](https://github.com/ahmeddx Fouad)
code and slides are found at this [github repo](#)