10 Object-Oriented Programming: Team Profile Generator

Your Task

Your task is to build a Node.js command-line application that takes in information about employees on a software engineering team, then generates an HTML webpage that displays summaries for each person. Testing is key to making code maintainable, so you'll also write a unit test for every part of your code and ensure that it passes each test.

Because this application won't be deployed, you'll need to provide a link to a walkthrough video that demonstrates its functionality and all of the tests passing. You'll need to submit a link to the video AND add it to the readme of your project.

> **Note**: There is no starter code for this assignment.

User Story

```
AS A manager
I WANT to generate a webpage that displays my team's basic info
SO THAT I have quick access to their emails and GitHub profiles
```

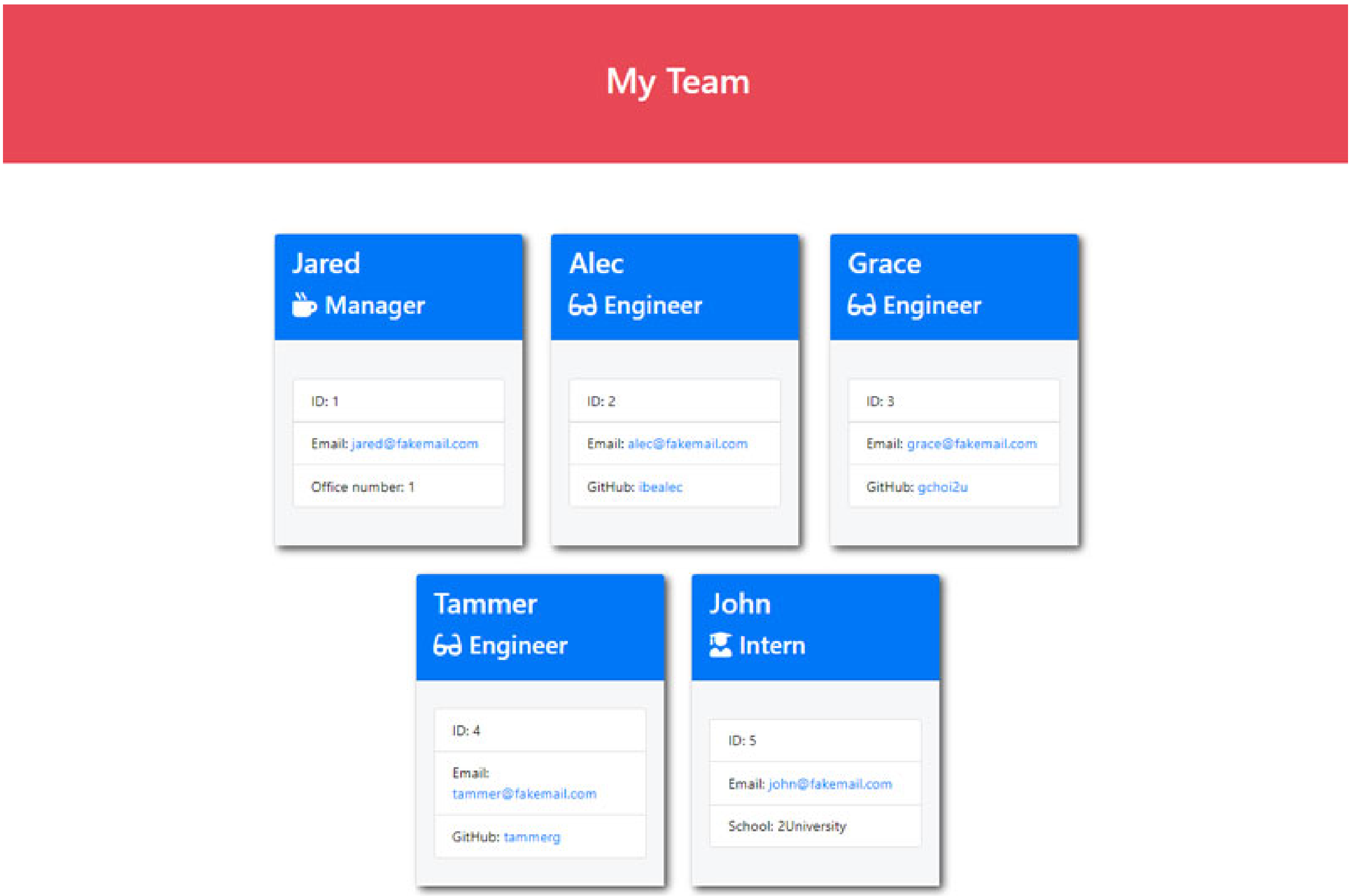
Acceptance Criteria

```
GIVEN a command-line application that accepts user input
WHEN I am prompted for my team members and their information
THEN an HTML file is generated that displays a nicely formatted team roster based on user input
WHEN I click on an email address in the HTML
THEN my default email program opens and populates the TO field of the email with the address
WHEN I click on the GitHub username
THEN that GitHub profile opens in a new tab
WHEN I start the application
THEN I am prompted to enter the team manager’s name, employee ID, email address, and office number
WHEN I enter the team manager’s name, employee ID, email address, and office number
THEN I am presented with a menu with the option to add an engineer or an intern or to finish building my team
WHEN I select the engineer option
THEN I am prompted to enter the engineer’s name, ID, email, and GitHub username, and I am taken back to the menu
WHEN I select the intern option
THEN I am prompted to enter the intern’s name, ID, email, and school, and I am taken back to the menu
```

```
WHEN I decide to finish building my team
THEN I exit the application, and the HTML is generated
```

Mock-Up

The following image shows a mock-up of the generated HTML's appearance and functionality:



My Team

Jared
Manager
ID: 1
Email: jared@fakemail.com
Office number: 1

Alec
Engineer
ID: 2
Email: alec@fakemail.com
GitHub: ibealec

Grace
Engineer
ID: 3
Email: grace@fakemail.com
GitHub: gchoi2u

Tammer
Engineer
ID: 4
Email: tammer@fakemail.com
GitHub: tammerg

John
Intern
ID: 5
Email: john@fakemail.com
School: 2University

The styling in the image is just an example, so feel free to add your own.

Getting Started

This homework will combine many of the skills we've covered so far. In addition to the User Story and Acceptance Criteria, we've provided some guidelines to help get started.

Your application should use Jest for running the unit tests and Inquirer for collecting input from the user. The application will be invoked by using the following command:

```
node index.js
```

It is recommended that you start with a directory structure that looks like the following example:

```
.
├── __tests__/             //jest tests
```

```
│   ├── Employee.test.js
│   ├── Engineer.test.js
│   ├── Intern.test.js
│   └── Manager.test.js
├── dist/                  // rendered output (HTML) and CSS style sheet
├── lib/                   // classes
├── src/                   // template helper code
├── .gitignore             // indicates which folders and files Git should ignore
├── index.js               // runs the application
└── package.json
```

Important: Make sure that you remove `dist` from the `.gitignore` file so that Git will track this folder and include it when you push up to your application's repository.

The application must include `Employee`, `Manager`, `Engineer`, and `Intern` classes. The tests for these classes (in the `_tests_` directory) must ALL pass.

The first class is an `Employee` parent class with the following properties and methods:

- `name`
- `id`
- `email`
- `getName()`
- `getId()`
- `getEmail()`
- `getRole()`—returns `'Employee'`

The other three classes will extend `Employee`.

In addition to `Employee`'s properties and methods, `Manager` will also have the following:

- `officeNumber`
- `getRole()`—overridden to return `'Manager'`

In addition to `Employee`'s properties and methods, `Engineer` will also have the following:

- `github`—GitHub username
- `getGithub()`
- `getRole()`—overridden to return `'Engineer'`

In addition to `Employee`'s properties and methods, `Intern` will also have the following:

- `school`
- `getSchool()`

- `getRole()`—overridden to return `'Intern'`

Finally, although it’s not a requirement, consider adding validation to ensure that user input is in the proper format.

Grading Requirements

This homework is graded based on the following criteria:

Deliverables: 15%

- A sample HTML file generated using the application must be submitted.

- Your GitHub repository containing your application code.

Walkthrough Video: 32%

- A walkthrough video that demonstrates the functionality of the Team Profile Generator and passing tests must be submitted, and a link to the video should be included in your README file.

- The walkthrough video must show all four tests passing from the command line.

- The walkthrough video must demonstrate how a user would invoke the application from the command line.

- The walkthrough video must demonstrate how a user would enter responses to all of the prompts in the application.

- The walkthrough video must demonstrate a generated HTML file that matches the user input.

Technical Acceptance Criteria: 40%

- Satisfies all of the preceding acceptance criteria plus the following:

  - Uses the Inquirer package.

  - Uses the Jest package for a suite of unit tests.

  - The application must have `Employee`, `Manager`, `Engineer`, and `Intern` classes.

Repository Quality: 13%

- Repository has a unique name.

- Repository follows best practices for file structure and naming conventions.

- Repository follows best practices for class/id naming conventions, indentation, quality comments, etc.

- Repository contains multiple descriptive commit messages.

- Repository contains a high-quality readme with description and a link to a walkthrough video.

Review

You are required to submit the following for review:

- A walkthrough video that demonstrates the functionality of the application and passing tests.

- A sample HTML file generated using your application.

- The URL of the GitHub repository, with a unique name and a readme describing the project.
