**Transition Academy**

**Stage 1 Sprint Topics and Outcomes**

(Note that the topics are at an introductory/foundational level. Any use of the topics in later stages should ideally be reviewed.)

|  |  |  |
| --- | --- | --- |
| **Sprint**  (2 days each) | **Topic** | **Outcomes** |
| 1 | HTML/CSS/Javascript | 1. Understand the meaning of full-stack development. 2. Understand the roles of the three foundational front-end technologies: HTML, CSS and Javascript. 3. Apply the three foundational technologies to create a front-end using a current IDE (VS Code). |
| 2 | Javascript and objects/classes; Source control | 1. Understand and use source/version control via git and Github. 2. Understand and use object-oriented concepts including objects and classes. 3. Learn and apply additional Javascript. |
| 3 | Javascript and APIs (Part I) | 1. Understand and use REST Application Programming Interfaces (APIs). 2. Understand and use JSON files. 3. Learn and practice applying REST APIs in Javascript. |
| 4 | Javascript and APIs (Part II) | 1. Further practice applying JSON files. 2. Further practice learning and applying REST APIs in Javascript. |
| 5 | Typescript (Part I) | 1. Apply Typescript to create a front-end application. |
| 6 | Typescript (Part II) | 1. Apply Typescript to create a front-end application. |
| 7 | Introduction to Angular Framework | 1. Describe the purpose of the Angular framework and the major elements of the framework. 2. Gain experience developing a front-end application using Angular through the completion of Angular tutorials. |
| 8 | Front-end Capstone | 1. Demonstrate a level of competency that allows you to create a front-end application using HTML, CSS, Javascript, Typescript, Angular (optional), VS Code, and git/Github. |
| 9 | Object-oriented Programming (OOP) w/ C#.NET (Part I) | 1. Understand the basic concept of object-oriented programming. 2. Create an application that uses programmer-defined classes and objects by applying the use of C#.NET in Visual Studio. |
| 10 | [OOP (Part II - Architectural patterns (MVC as a layered pattern))](https://lms.dontpaniclabs.com/courses/15/modules/items/3067) | 1. Describe modern architectural principles and patterns. 2. Understand MVC as an architectural pattern that helps achieve separation of concerns. 3. Create an application that uses the MVC architectural pattern using C#.NET in Visual Studio. |
| 11 | [(Part III - Interfaces and related topics (IoC and DI))](https://lms.dontpaniclabs.com/courses/15/modules/items/3068) | 1. Understand interfaces. 2. Describe topics related to interfaces including inversion of control (IoC) and Dependency Injection (DI) 3. Create an application that implements interfaces and dependency injection using C#.NET in Visual Studio. |
| 12 | [OOP (Part IV - Unit testing and SOLID principles)](https://lms.dontpaniclabs.com/courses/15/modules/items/3069) | 1. Understand unit testing options using C#.NET in Visual Studio 2. Describe best practices in OOP as captured in SOLID. 3. Create unit tests for applications that using C#.NET in Visual Studio. |
| 13 | [Full-stack web app (w/ .NET Web APIs and Entity Framework)](https://lms.dontpaniclabs.com/courses/15/modules/items/3070) | 1. Understand the process of creating Web APIs in .NET 2. Understand the basics of the Entity Framework 3. Create a full-stack web application that uses .NET Web APIs and the Entity Framework using C#.NET Visual Studio. |
| 14 | Introduction to Node.js and Express | 1. Understand where NodeJS and Express fit into the full-stack ecosystem 2. Create applications (via tutorials) that use Node.js and Express. |
| 15 | Stage 1 Capstone | Demonstrate a level of competency that allows you to create a full-stack application using several of the following: HTML, CSS, Javascript/Typescript, C#, Web APIs in .NET, Entity Framework, Node.js w/ Express, Visual Studio and/or Visual Studio Code, Postman and git/Github. |