# Introduction

The purpose of this report is to outline the development process undertake to deploy a functional public-facing web application for a fictional sporting organisation named the Penguins Basketball Club. This report will outline the methodology undertaken to develop, test and deploy the web application and conclude with a reflection on the software development process.

# Methodology

## Planning

This development project will follow the planning sequences outlined in the standard development lifecycle (SDLC) (ISO 2015) and utilise the Agile project planning method (Agile Alliance 2023). Development work will be structured to occur within weeklong sprint periods each focusing on different development requirements as outlined below:

* Week 1 – React Development and Version Control
* Week 2 – Code Refactoring and Error Handling
* Week 3 – Testing Driven Development
* Week 4 – Deployment and Maintenance

## Development

The development of the functional website application will be achieved via React. React will be used due to its modular architecture that allows for enhanced flexibility in the application's structure in addition to its ease of maintenance (A 2019). Components within the web application will be divided into the various functions within the site. Individual components will be created for each section of the web application. Carlson et al. (2003) state is highly important that the website incorporate interactive elements enabling sports fans to immerse themselves into the website, creating stronger fan identification with the team, resulting in potential revenue generation. As such, third party packages will be used where possible to enrich the user experience. A summary of each component is provided in the table below:

***Table 1*** *App components*

|  |  |  |
| --- | --- | --- |
| **Component** | **Description** | **Jira Issue** |
| *Navigation* | Represents the navigation bar of the web application. Provides following functionality:   * Includes links to various page sections stored within array. * Renders mobile menu on small devices. * Includes third party package ‘React-Scroll’ (F 2023) to enhance navigation performance. |  |
| *Logo* | Includes the site logo. Stored in own component due to reuse in multiple sections. |  |
| *Main* | Represents the body of the web application and stores components of individual page sections. See Table 2 below. |  |
| *Footer* | Represents footer of the web application includes following functionality:   * Page Scroll to top function when reach page end via logo click. * Reuses logo component |  |
| *Error Boundary* | Class function to recognise errors rendering in web application. Couches components above to communicate errors to end users. |  |

***Table 2*** *Main Components*

|  |  |  |
| --- | --- | --- |
| **Component** | **Description** | **Jira Issue** |
| *Home* | Includes following functionality:   * Third party package ‘React-responsive-carousel’ (L 2022) utilised to add interactivity * Includes div change on click event |  |
| *About Us* | Simple block text outlining club mission statement. |  |
| *Schedule* | Includes following functionality:   * Game schedule stored in array * Filter applied to game schedule to show only game dates greater than today * Function to build game schedule card * Renders mapped game schedule array in date order * Search function applied to array to enhance user functionality * Includes functionality to download team schedule document |  |
| *Roster* | Includes following functionality:   * Third party package react-card-flip (A 2022) utilised to enhance user interactivity. * Team player information stored in array * Function to build individual player card rendering image and player statistics as table * Renders roster section in full including mapped array to show all player details. |  |
| *Standings* | Table to store all team season progress.   * Stores team details in array * Mathematic functions applied to determine additional statistics * Renders row for each team inclusive of stats * Orders table rows on win total * Retrieves component that stores team iconography matched by team name |  |
| *Community* | Section to include sponsor details and mailing list sign up.  Team sponsors:   * Team sponsors stored as individual components to improve site management * Description of sponsors includes test string limit to improve site presentation * Team sponsors rendered as individual cards   Mailing list:   * Input form utilised to drive fan engagement * Functions utilised to download privacy and terms policies * Function utilised to display received message to user on submission event inclusive of template string. |  |
| *Gallery* | Grid view of embedded videos rendered to end user. Little additional .  Functionality. |  |
| *Contact* | Section to store contact details of team. Includes iframe stored as separate component to better manage location changes. |  |

## Version Control

Version control for this project will be managed via Git due to its ease of access to the developer. The code was hosted on a GitHub repository, allowing for easy sharing of the codebase with other developers working on the project. Initial changes to the code base have been made via pushes to the master branch. Further code maintenance to manage ongoing maintenance will be accomplished via branching.

## Deployment

To ensure the application was functioning as intended, a testing framework was used to write unit and integration tests for the codebase.

This ensured that any code changes were tested before they were merged into the codebase, reducing the risk of introducing bugs or errors.

Due to the small scale of the web application, and the developers limited experience in testing automation, a Continuous Integration (CI) strategy was utilised for deployment.

Deployment was managed by the site author via a deployment checklist.

The deployment platform used for this project was GitHub Pages (2023). The finalised web product is deployed at <https://lwturnbull.github.io/PenguinsBasketballWebsite/>.

# Reflection

Babb et al. (2014) recognise that without ongoing reflection, software development teams can lose their ability to learn and improve. It is therefore necessary to reflect upon the actions undertaken in this project to improve the quality of the software development lifecycle and to identify opportunities for further improvement.

The first sprint in this project required the rewriting of the initial website prototype into a React based project. I found that by structuring the site into components made for managing changes significantly easier, however I experienced some difficulties in understanding the subtle differences in React in comparison to HTML. This sprint also introduced the concept of version control management via Git. I had not properly considered the requirements for version management in the planning stages for this project. This contributed to a greater number of commits than would have otherwise been necessary in a more structured project. This might make interpretation by other users more challenging which could impede project success (Ernst 2012). Although I would mostly attribute this my approach to my lower level of knowledge, I would take care in future to stage commits and develop the site in sections to create a more secure, leaner code base.

The second sprint in this project required code necessitated some code refactoring, the introduction of error handling and the use of third-party packages. I found these activities to be highly useful for my own personal development, especially in recognising opportunities to improve the user experience. I attempted to identify several third-party packages that would enable functionality currently beyond my current level of experience and provide interactivity as this would benefit the product solution (Carlson et al. 2003). I struggled in finding opportunities to better utilise error handling protocols and error boundaries. This contributed to my project likely underutilising this functionality. I believe that through further experience and reading that I can overcome this issue.

The third sprint of this project focussed on testing driven development (TDD). I found this to be the most challenging component of the development project as I lacked understanding in what I should be testing for and how to protect against vulnerabilities. Whist I was able to implement some simple automated testing to check the rendering of specific page elements, the more complicated functions of user behaviour were relegated to manual user-based testing. This also proved challenging as it is difficult to ensure objectivity when developing and testing are completed by the same groups (Shah 2020). Regardless, I would like to work towards better understanding different testing tools and standard so that I might improve the reliability and security of my approach to development.

The fourth and final sprint of this project was focussed on deployment and maintenance. I found that the deployment process itself was straight forward, however I would likely allot more time to planning deployment than I had for this project. I believe this challenge was resultant from my learning the steps of deployment parallel to the project development. I might also attempt to utilise a continuous deployment strategy, however due to the constraints of user availability within the development of this project, this would have proven unsuitable (Alvatar 2017).

To conclude, creating a web application has been a challenging yet rewarding experience. Throughout the process, I have gained a deeper understanding of web development and the importance of user-centric design. Moving forward, I hope to continue to hone my skills and overcome challenges as I progress throughout my career.

# References

A. (2022), ReactCardFlip, npmjs, viewed 25 March 2023, <https://www.npmjs.com/package/react-card-flip>

A, V (2019), ‘Top 10 Advantages of using React.JS’, Medium, viewed 25 March 2023, <https://medium.com/@cs.venkatesh95/top-10-advantages-of-using-react-js-aea928548aef>

Agile Alliance (2023), ‘What is Agile?’, viewed 5 February 2023, <https://www.agilealliance.org/agile101/ >

Altvatar, A (2017), ‘Dev leaders compare continuous delivery vs. continuous deployment vs. continuous integration’, Stackify, viewed 1 April 2023, <https://stackify.com/continuous-delivery-vs-continuous-deployment-vs-continuous-integration/Links to an external site.>

Babb, J, Hoda, R and Nørbjerg, J (2014), ‘Embedding reflection and learning into agile software development’, IEEE software, vol. 31 (4), pp. 51-57.

Carlson, J, Rosenberger, P, & Muthaly, S (2003), ‘Nothing but net! A study of the information content in Australian professional basketball websites’, Sport Marketing Quarterly, 12 (3), p.184-188.

Ernst, M (2012), ‘Version control concepts and best practices’, University of Washington, viewed 1 April 2023, <https://homes.cs.washington.edu/~mernst/advice/version-control.html>

F (2023), React Scroll, npmjs, viewed 25 March 2023, <https://www.npmjs.com/package/react-scroll>

Font Awesome (2023), Font Awesome, viewed 12 February 2023, <https://fontawesome.com/>

Git (2023), Git, viewed 18 March 2023, <https://git-scm.com/>

GitHub Pages (2023), GitHub, viewed 18 March 2023, < https://pages.github.com/>

ISO (2015), ISO/IEC/IEEE 23026:2015(en), viewed 5 February 2023, <https://www.iso.org/standard/62440.html>

J. (2020), ‘React-Testing-Library – Pro tips’, Medium, viewed 25 March 2023, <https://jero786.medium.com/react-testing-library-pro-tips-eba7181eb6fb>

Koutsabasis, P, Vlachogiannis, E, & Darzentas, J S (2010), ‘Beyond specifications: Towards a practical methodology for evaluating web accessibility’, Journal of usability studies, 5(4), p.157-171

L. (2022), React Responsive Carousel, npmjs, viewed 25 March 2023, <https://www.npmjs.com/package/react-responsive-carousel>

Privacy Policy Template (2023), Victorian Government, viewed 12 February 2023, <https://business.vic.gov.au/tools-and-templates/privacy-policytemplate>

Shah, D (2020), ‘Should developers do their own testing?’, Medium, viewed 28 March 2023, <https://medium.com/tenets/should-developers-test-their-own-code-c0a921de1a9c>

Uizard (2023), Uizard, viewed 5 February 2023, <https://app.uizard.io/p/b0312edf>

Unsplash (2023), Unsplash, viewed: 12 February 2023, <https://unsplash.com/>

# Appendix 1 – Testing Checklist

# Appendix 2 – Automated Testing

|  |  |
| --- | --- |
| **Automated Test** | **Description** |
| import React from 'react';  import {render, screen, fireEvent} from '@testing-library/react';  import {axe, toHaveNoViolations} from 'jest-axe'    import Home from './Components/Main/Home/Home';    //Test to confirm image carousel includes valid src attributes    describe('Home', () => {      test('renders all carousel images', () => {        render(<Home />);        const carouselImages = screen.getAllByAltText(/carouselbackground\d/i);        carouselImages.forEach((image) => {          expect(image).toHaveAttribute('src');          expect(image).toBeVisible();        });      });    }); | The Penguins Basketball team require that their home drive player and team interest. They have chosen to accomplish this via the inclusion of a scrolling news carousel that welcomes users as they visit the site. This section should render images to support each article. The purpose of this test is to confirm that the images rendered in this section are visible using the ‘jest’ testing library. This test should pass as per below prior to deployment: |
| import React from 'react';  import {render, screen, fireEvent} from '@testing-library/react';  import {axe, toHaveNoViolations} from 'jest-axe'    import AboutUs from './Components/Main/AboutUs/AboutUs';  //test About Section includes reference to club.  test('renders Penguins about', () => {    render(<AboutUs />);    const linkElement = screen.getByText(/Penguin/i);    expect(linkElement).toBeInTheDocument();  }); | The Penguins Basketball team website should act as a primary knowledge base for its users. This test checks whether the "AboutUs" component renders properly and includes the word "Penguin" in the block text of the club mission statement. This test should pass as per below prior to deployment: |

|  |  |
| --- | --- |
| import React from 'react';  import {render, screen, fireEvent} from '@testing-library/react';  import {axe, toHaveNoViolations} from 'jest-axe'  import Roster from './Components/Main/Roster/Roster';  describe('Roster', () => {    test('renders all team roster images', () => {      render(<Roster />);      const rosterImages = screen.getAllByAltText(/Player/i);      rosterImages.forEach((player) => {        expect(player).toHaveAttribute('src');        expect(player).toBeVisible();      });    });  }); | The Penguins Basketball team website should act as a primary knowledge base for its users. The team roster should be displayed at all times including pictures of the players to drive fan recognition and engagement. This test aims to find all image components as rendered in the roster component and checks if the img attribute to be visible using the ‘jest’ testing library. This test should pass as per below prior to deployment: |
| import React from 'react';  import {render, screen, fireEvent} from '@testing-library/react';  import {axe, toHaveNoViolations} from 'jest-axe'  import Schedule from './Components/Main/Schedule/Schedule';  // tests render of schedule  describe('Schedule', () => {    test('renders Schedule component', () => {      render(<Schedule />);    });  }); | The Penguins Basketball team website should act as a primary knowledge base for its users. It is critical that the team schedule always remain accessible. The schedule is comprised of an array of individual game details that can be stored at high volume. This test checks whether the "Schedule" component can be rendered without crashing. This test should pass as per below prior to deployment: |

|  |  |
| --- | --- |
| import React from 'react';  import {render, screen, fireEvent} from '@testing-library/react';  import {axe, toHaveNoViolations} from 'jest-axe'  import Standings from './Components/Main/Standings/Standings';  //test render of standings table  describe('Standings Table>', () => {    test('renders a table', () => {      render(<Standings />);      const tableElement = screen.getByRole('table');      expect(tableElement).toBeInTheDocument();    });  }); | The Penguins Basketball team website should act as a primary knowledge base for its users. It is critical that the team standings always remain accessible. This test checks whether the "Standings" component renders a table properly by searching for a table element in the rendered output and checking whether it is present in the document. This test should pass as per below prior to deployment: |
| import React from 'react';  import {render, screen, fireEvent} from '@testing-library/react';  import {axe, toHaveNoViolations} from 'jest-axe'    import MailingListForm from './Components/Main/Community/MailingList';  //test mailing list form rendered correctly  describe('Mailing List Component', () => {    test('is accessible to all', async () => {      expect.extend(toHaveNoViolations);      const {container} = render(<MailingListForm />);      const results = await axe(container);      expect(results).toHaveNoViolations();    });  }); | The Penguins Basketball team require that their mailing list form is generally accessible for all users. This function tests whether the "MailingListForm" component is accessible to all users by using the "axe" accessibility testing library to check for any accessibility violations, and then asserting that there are no violations. This test should pass as per below prior to deployment: |

|  |  |
| --- | --- |
| import React from 'react';  import {render, screen, fireEvent} from '@testing-library/react';  import {axe, toHaveNoViolations} from 'jest-axe'  import MailingListForm from './Components/Main/Community/MailingList';  //test form submission  describe('MailingList Component', () => {    test('form message sent', () => {      render(<MailingListForm />);      window.alert = jest.fn();      const nameInput = screen.getByPlaceholderText('Enter your name');      const emailInput = screen.getByPlaceholderText('Enter your email');      const submitButton = screen.getByText('Submit');      fireEvent.change(nameInput, { target: { value: 'John Doe' } });      fireEvent.change(emailInput, { target: { value: 'john.doe@example.com' } });      fireEvent.click(submitButton);      expect(window.alert).toHaveBeenCalledWith('Welcome to the team John Doe! Keep an eye on your inbox for exciting news and developments.');    })  }); | The Penguins Basketball team require that their mailing list form send a message to users upon submission. This test checks whether the "MailingListForm" component correctly sends a welcome message after the user submits their name and email by simulating user interactions with the form inputs and submit button using the "fireEvent" utility from the "@testing-library/react" library. The test then checks whether the expected message is displayed in an alert using the "toHaveBeenCalledWith" matcher on the mocked "window.alert" function. This test should pass as per below prior to deployment: |

|  |  |
| --- | --- |
| import React from 'react';  import {render, screen, fireEvent} from '@testing-library/react';  import {axe, toHaveNoViolations} from 'jest-axe'  import Gallery from './Components/Main/Gallery/Gallery';  // test to check all videos contain valid src attributes  describe('Gallery Component', () => {    test('only contains active iframe src links', async () => {      render(<Gallery />);      const iframes = await screen.findAllByTitle(/youtube video/i);      iframes.forEach((iframe) => {        const frame = document.createElement('iframe');        frame.src = iframe.src;        frame.onload = () => {          expect(frame.contentWindow.location.href).not.toBe('about:blank');        };        document.body.appendChild(frame);      });    });  }); | The Penguins Basketball team require that their website provide users a means to access media content including videos. The purpose of this test is to check all of the video elements rendered within the Gallery component using an iframe and whether the src attribute used includes an active link. This will help to ensure that content is not overridden that may negatively affect the user experience. This test should pass as per below prior to deployment: |
| import React from 'react';  import {render, screen, fireEvent} from '@testing-library/react';  import {axe, toHaveNoViolations} from 'jest-axe'    import Map from './Components/Main/Contact/Map';  //test contact iframe  describe('Contact Component', () => {    test('only contains iframe', () => {      render(<Map />)      const iframeElement = screen.getByTitle('Map Location Embed');      expect(iframeElement).toBeInTheDocument();    });  }); | The Penguins Basketball team require that their contact details remain present on the site and that this includes address details rendered via a map. This test checks whether the "Map" component only contains the expected iframe element by searching for an iframe element with the specific title "Map Location Embed" and asserting that it is present in the rendered output. This test should pass as per below prior to deployment: |