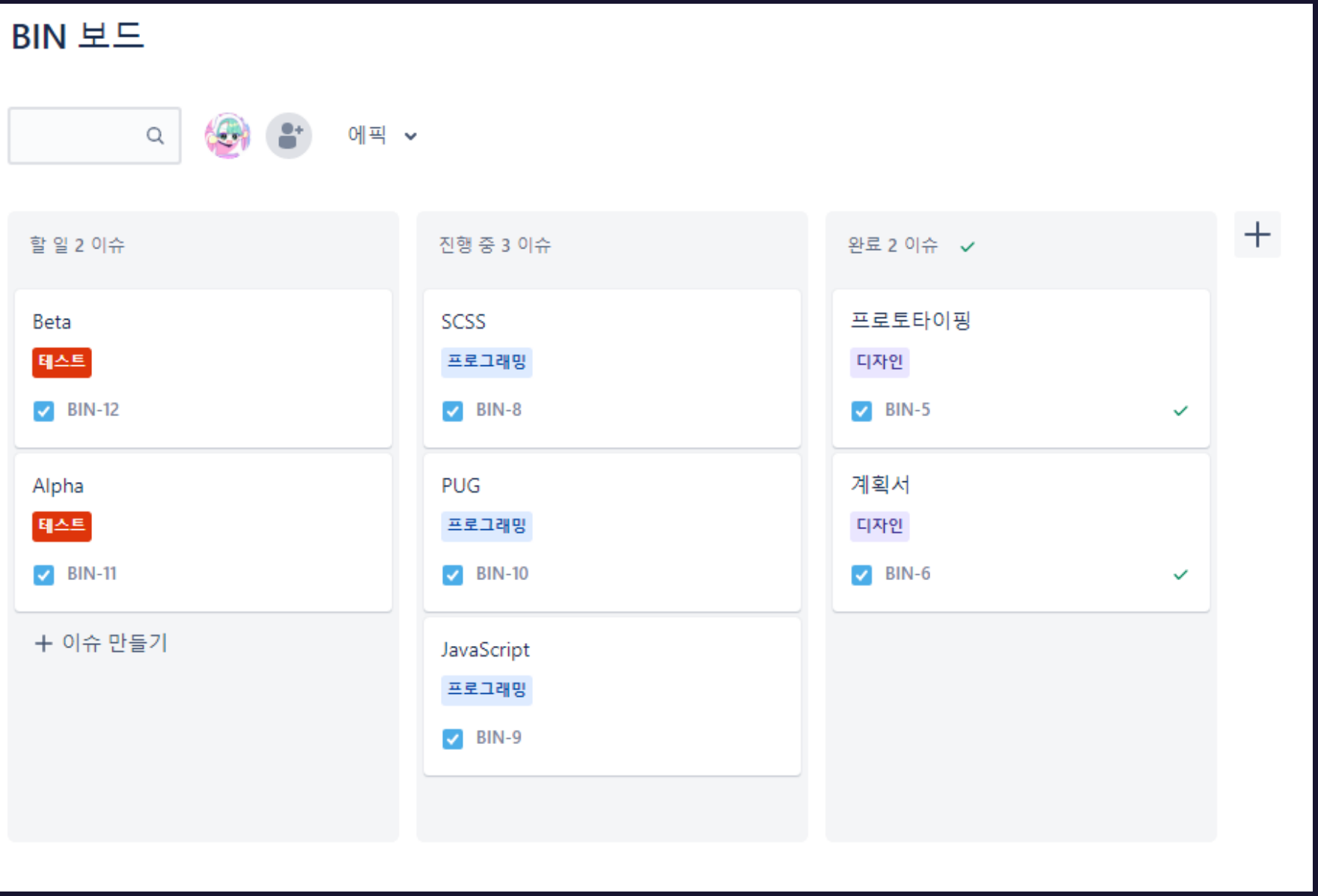
진수 변환기

라인컴퓨터아트학원 / JAVA기반 스마트웹 디지털컨버젼스 17기 / 서동민

목차

계획표

링크



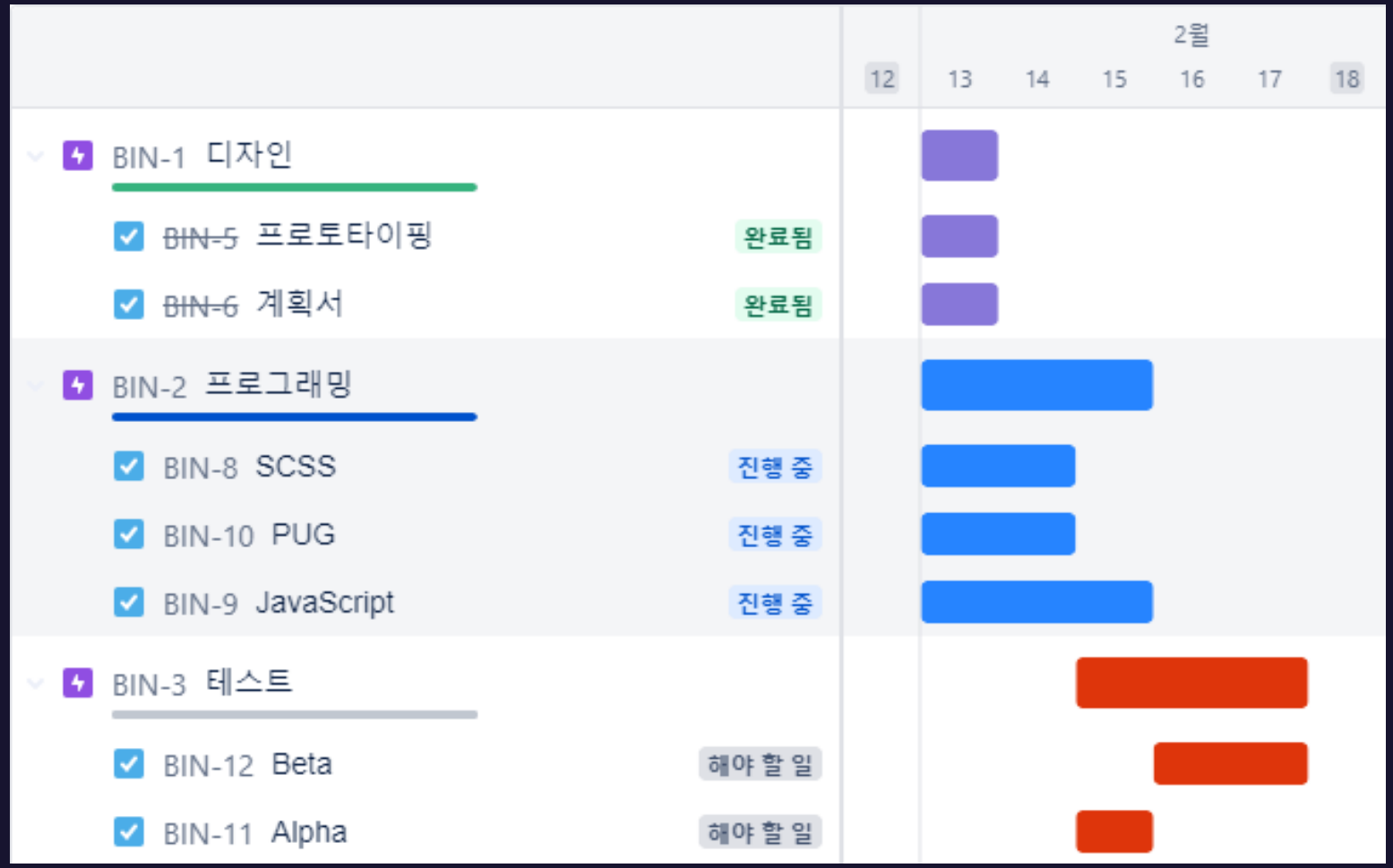
스크럼 보드를 이용해서 진행도를 정리하였다.

해야 할 일, 진행 중, 완료됨으로 분류했다.

스크럼 보드를 이용하여 진행상황을 한눈에 확인할 수 있어서 진행에 도움이 되었다.

계획표

링크



간트 차트를 이용해서 일정을 새웠다.

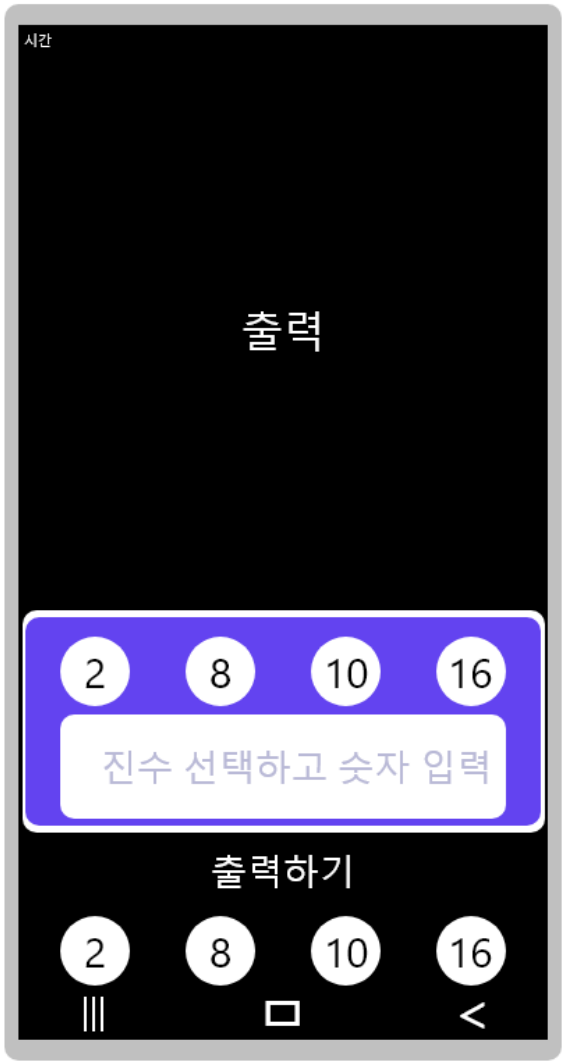
대분류로는 디자인, 프로그래밍, 테스트로 나누어 계획했다.

디자인은 계획서 작성과 프로토타이핑 제작으로 계획했다.

프로그래밍은 틀은 PUG, 디자인은 SCSS, 내부 동작 구현은 JavaScript로 계획했다.

프로토타이핑

링크



디자인 모티브는 갤럭시 노트20 울트라를 기준으로 했다.

프로그램의 동작을 구현하기 위해서 사용자의 입력 행위를 다음과 같이 생각했다.

1. 입력할 값의 진수를 선택한다.
2. 값을 입력한다.
3. 출력할 진수를 선택한다.
4. 출력 부분에 숫자가 출력된다.

위 순서대로 입력이 진행된다면 필요한 입력이 몇 개인지 생각하고 디자인을 진행했다.

직관적인 인터페이스를 구성하려고 했지만 사용자가 인터페이스를 이해하지 못할 수도 있기 때문에 숫자를 입력하는 부분에 무엇을 입력해야 하는지 표기했다.

프로그래밍

링크

1

PUG

간편하고 단결한 문법을 가졌다.

퍼그를 통해 HTML 문서를 더욱 빠르게 작성할 수 있었다.

2

SCSS

효율적인 디자인을 할 수 있다.

SCSS를 이용하여 변수, Mixin 같은 기능을 통해서 효율적이고 수정이 편한 디자인을 구현할 수 있었다.

3

JavaScript

동작 구현이 가능하다.

동작들을 구현하기 위해서 사용했다. Class를 이용한 구조를 통해서 효율적인 형태와 기능을 구현할 수 있었다.

PUG

링크

```
body
    #fingerPrint.xi-fingerprint
    img.clip(src="../img/clip.png", alt="클립")
    img.note(src="../img/note.png", alt="노트")
    img.pen(src="../img/pen.png", alt="펜")
    img.eraser(src="../img/eraser.png", alt="지우개")
    main
      .radiusBlock
        .topTool
          span.timeTool
          .batteryTool.xi-battery
          .wifiTool.xi-rss
        .xi-spinner-3.xi-spin.loading
        output
```

PUG를 이용하여 기본적인 틀을 잡았다.

PUG는 HTML문서처럼 태그를 열고 닫을 필요가 없어서 제작하는데 많은 시간을 줄일 수 있었다.

div에 class나 id를 부여하는 경우에는 div라는 것을 명시해줄 필요가 없어서 빠른 진행을 할 수 있었다.

들여쓰기로 태그의 범위를 구분하는 특성 때문에 같은 수많은 닫는 태그들로 구분하는 것이 아닌 단순한 위치 비교로 태그의 범위를 구분할 수 있기 때문에 태그의 위치를 확인하는데 큰 도움이 되었다.

SCSS

링크

```scss
main {
    .radiusBlock {
      border-radius: 10/1920 * 100vw;
      background-image: linear-gradient(
        to left,
        #555 0%,
        #000 5%,
        #000 95%,
        #555 100%
      );
      visibility: visible;
      opacity: 1;
      overflow: hidden;
    }
    //min-width: 400/1920 * 100vw;
    min-height: 650/1920 * 100vw;
    width: 400/1920 * 100vw;
    //height: 650/1920 * 100vw;
    position: absolute;
    top: 50%;
    left: 50%;
```

SCSS를 이용하여 디자인을 하였다. SCSS는 @mixin과 변수, 함수 등을 이용할 수 있었다.

SCSS의 편한 괄호 안에 자식 태그를 적어서 스타일을 적용할 수 있는 방법 덕분에 스타일을 주기 위해서 선언하는 클래스의 수가 줄었다.

연산기능 덕분에 vw에 대한 수치를 일일이 계산해서 대입해주는 불필요한 일을 줄일 수 있었다.

SCSS의 괄호 안에 태그를 표기할 수 있는 방법 덕분에 스타일이 겹치거나 중복으로 효과가 적용되는 경우가 줄었다.

스타일의 위치를 찾는 것이 매우 편해졌다.

JavaScript

링크

```
class BinaryCalculator {
  //! 필드 ES13 부터 지원 중이나 정식 지원단계가 아님
  constructor() {
    this.p_takeInputValue = 0;
    this.p_takeInputBinary = 0;
    this.p_outputValue = 0;
    this.p_outputBinary = 0;
    this.p_takeInputValueToBinary10 = 0;
  }
  set setTakeInputBinary(takeInputBinary) {
    this.p_takeInputBinary = takeInputBinary;
  }
  set setTakeInputValue(takeInputValue) {
    this.p_takeInputValue = takeInputValue;
  }
  get getTakeInputValue() {
    if (this.p_takeInputValue != "") {
      return this.p_takeInputValue;
    }
  }
```

진수 변환기를 구성하는 오브젝트와 진수 변환기의 동작을 담당하는 기능들을 Class로 구현하였다.

생성자 함수에는 입력 값, 입력 진수, 출력 값, 출력 진수, 입력한 값을 10진수로 변환한 값을 담고 있는 변수를 클래스 변수로 생성했다.

```
const calculatorClass = new BinaryCalculator();
calculatorClass.eventWatcherInput();
calculatorClass.eventWatcherOutput();
calculatorClass.windowClick();
calculatorClass.fingerPrintTouch();
```

객체를 생성하고 입력창에 입력되는 값의 길이나 형식을 감시하는 객체의 메소드를 호출하였다.

JavaScript JS

Switch case 문을 이용하여 들어온 숫자 중에서 10 이상의 숫자를 문자로 변환하는 구조를 이용하여 시각적으로 보기 좋은 구조를 구현하였다.

```
inputNumberToDecimal() {
    switch (this.p_takeInputBinary) {
      case "2":
        this.event2To10();
        break;
      case "8":
        this.event8To10();
        break;
      case "10":
        this.setTakeInputValueToBinary10 = this.getTakeInputValue;
        break;
      case "16":
        this.event16To10();
        break;
    }
  }
```

```
eventWatcherInput() {
    this.p_regex2 = /[^01]/g;
    this.p_regex8 = /[^0-7]/g;
    this.p_regex10 = /[^0-9]/g;
    this.p_regex16 = /[^0-9a-fA-F]/g;
```

정규식을 이용하여 입력할 수 있는 숫자와 문자를 제한하였다.

각 진수의 구성에 들어갈 수 없는 숫자나 문자는 자동으로 제거하는데 사용되었다.

링크

결과물

링크

버그를 찾고 버그에 대한 해결 방법으로 입력을 제한하는 코드를 추가했다.

해결하지 못한 문제들을 이후 테스트하면서 발견하고 지속적으로 해결했다.

테스트를 진행하며 배경이 밋밋하다는 의견을 수용하여 배경에 나무의 질감이 반복되는 이미지를 활용했다.

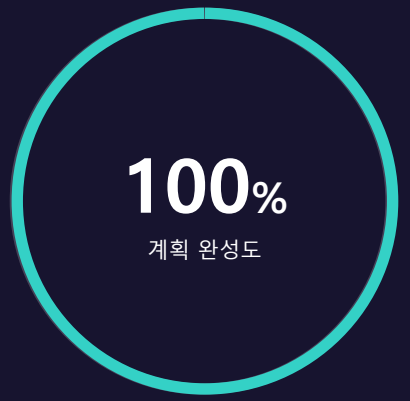
리뷰

링크

프로토타이핑

아쉬움이 많이 남았다.

더 많은 요소에 디자인이 필요했다고 생각된다.



계획

오차 없는 계획 그리고 많은 경험을 얻었다.

계획표에 계획한대로 오차 없이 프로젝트가 진행되었다. 스크럼 보드의 경우 좀 더 세분화 했으면 좋았을 것 같다고 느꼈다.

JavaScript

객체지향 프로그래밍

객체지향 프로그래밍을 통한 프로그래밍

처음 배운 객체지향이었는데 생각한데로 동작했다.

SCSS

디자인 감각이 성장했다.

디자인은 이전에 비해서는 성장했다고 느꼈다.

앞으로 계속 성장할 수 있을 것이라고 생각한다.