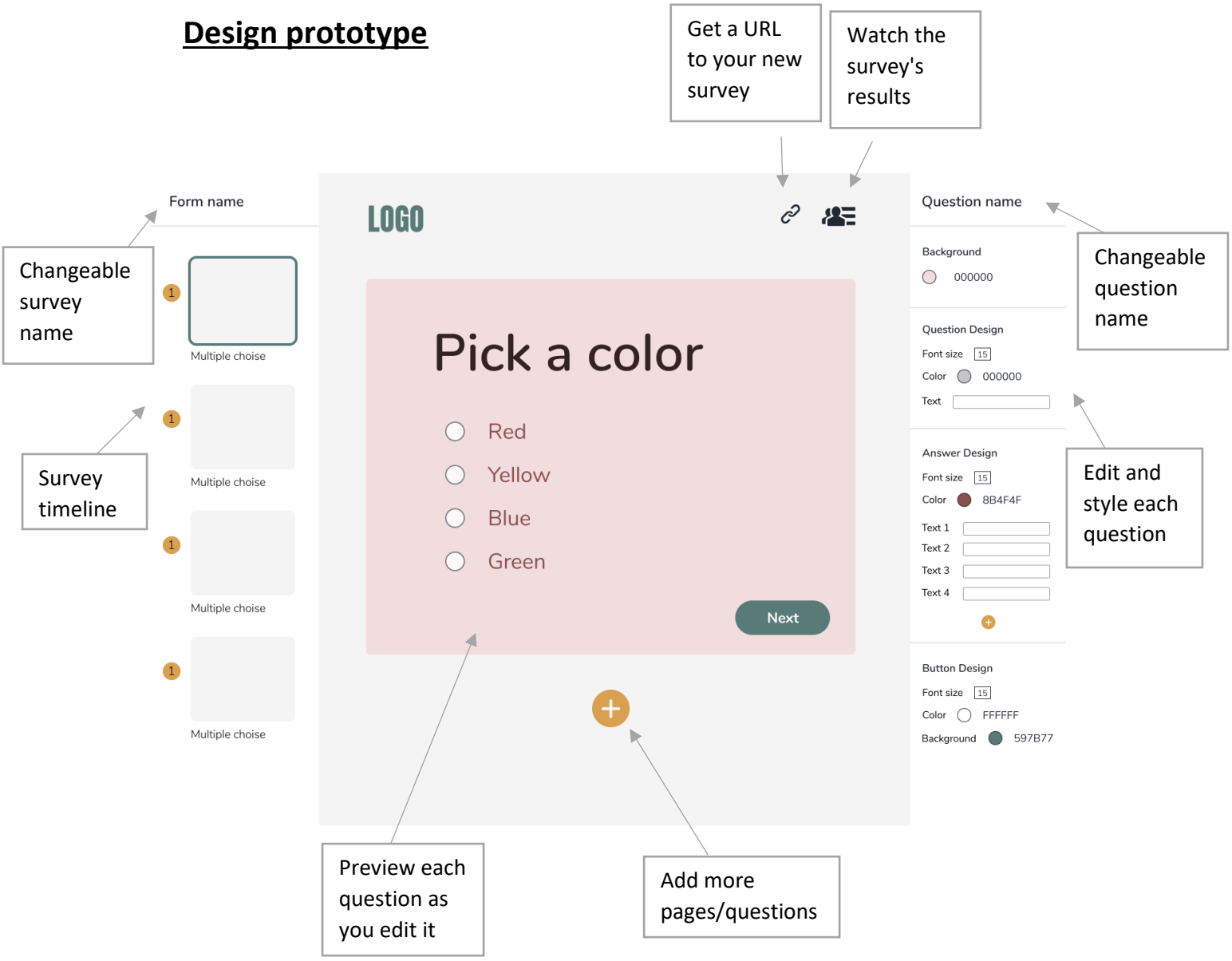
Specification Document- SurveyBox

SurveyBox is a browser-based application for creating, editing and viewing surveys.

Front end

Design prototype



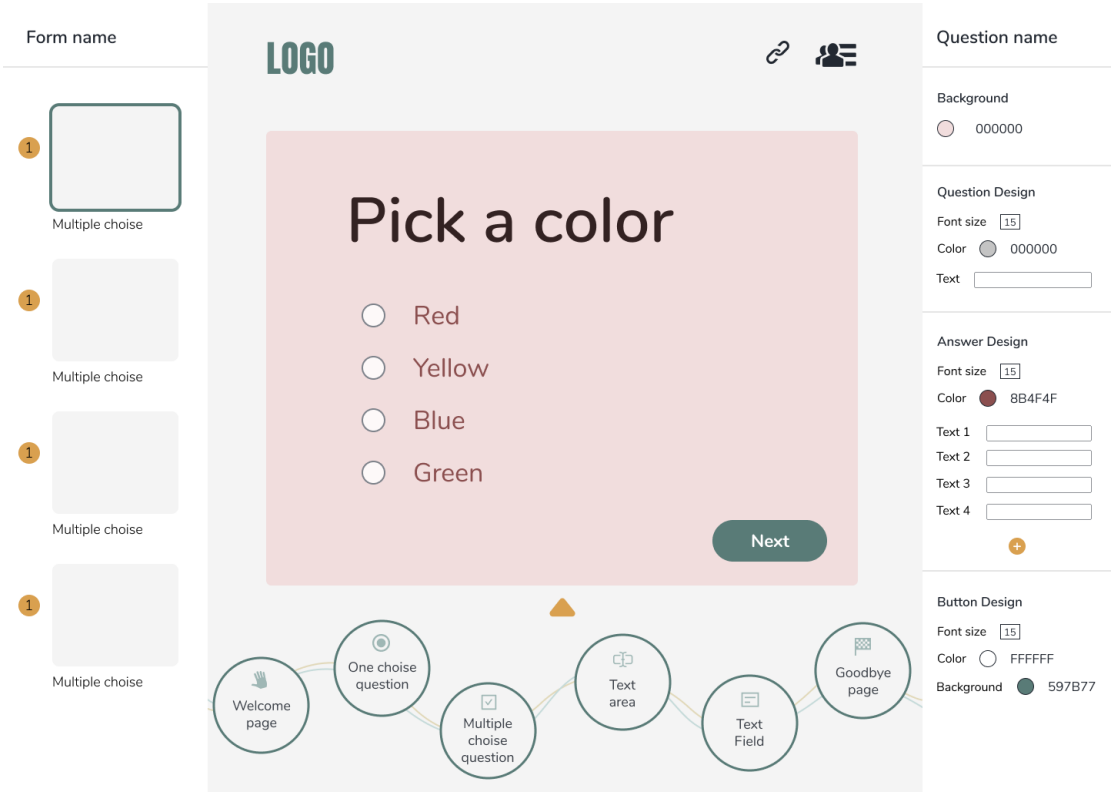
Code design

This project will be written in TypeScript using React and Redux libraries.

The UI is designed by me and will be implemented using CSS and react syntax HTML.

The UI is divided to components like so:

1. Left side bar- displays the survey's timeline: their order, type, and preview (will be implemented later on). Also has the option to change the survey's name.
   The current displayed question is marked with a distinct border.
2. Right side bar- displays the styling and editing options of the current question.
3. Header bar- displays the logo and 2 options:
   - copy this survey's URL
   - copy the results page URL
4. preview window- a preview of the current question. Previews the changes made in the edit bar immediately.
   This window is not interactable.
5. Add button- when clicked, displays the pages available to add.
   Like so:



Notes

- The first implementation of the project will focus on one survey only (one URL).
- There won't be a registration/login process at this point.

- The results will be the last feature to implement- if the survey is edited, we need to check if the results are no longer relevant and delete them. Every user can answer the survey multiple times because there is no registration at this point.

Survey's JSON object

each page object/component will implement the interface- IPage.

In addition, they will implement the interface "IStatement" or the interface "IQuestion".

One more interface is "INextPage", implemented by all pages except goodbye page, which is the last page.

Each interface has its own unique properties, regarding their edit and style options.

IPage

```
interface IPage {
  pageType: string;
  background: string;
  pageName: string;
}
```

IQuestion

```
interface IQuestion {
  questionType: string;
  questionColor: string;
  questionFontSize: number;
  questionText: string;
  answersColor: string;
  answersFontSize: number;
  answersText: Array<string>;
}
```

IStatementPage

```
interface IStatementPage {
  headerFontSize: number;
  headerColor: string;
  headerText: string;
  subHeaderFontSize: number;
```

```
  subHeaderColor: string;
  subHeaderText: string;
}
```

INextClick

```
interface INextClick {
  buttonFontSize: number;
  buttonBackground: string;
  buttonTextColor: string;
}
```

Events

There are 2 events "category" in this app. Each event fires a dispatch and updates the state accordingly.

1. Page changes- those are events fired from the left toolbar (timeline) or from the page options (add new page buttons). Some examples are:
   - new page click event
   - Delete page click event
   - Rearrange pages (drag and drop event)
   - Page name changed
   - Survey name changed
   - current page index changed click event
2. design/edit changes- those events are fired from the edit toolbar (right side toolbar). Some examples are:
   - background changed
   - font size changed (question/answer/header/sub header)
   - text changed (question/answer/header/sub header)

back end

the server side is going to be written in Javascript using node.js and Express.

I will also use MongoDB and Mongoose to store the data.

As a first milestone, i will focus on one survey only- therefore, one URL.

Saving to the database

Each dispatch in the client side (which indicates on some change in the survey) will make the necessary updates and then send a patch request to the server, with the relevant changes, which will update the database accordingly.

"SurveyModel" will be a model representing all the survey's required and optional properties.

When a new page is added, the client side create an object of the relevant page. Then, the new page object will be sent to the server side, which will generate a "SurveyModel" object with the request object data, and save it to the database.

MongoDB will generate a unique id for this survey object, which will be the URL identifier.

Loading the interactable survey URL

The URL will be something like this: "http://www.surveybox.com/123456"

On page load, the client side will send a get request with the URL id parameter (i.e., 123456). The server will search the database for a survey with that id and send it as JSON (if it exist).

Then, the client side will go through the object properties and generate the survey pages accordingly.

Each survey page will have a URL of this format:
"http://www.surveybox.com/<survey id>/<current page number>"

The "next" button on click event will fire a post request with this page answer (if valid and not a welcome page with no question in it). The server will update the answer in the database.

* One thing to pay attention to- if the user changes the answer options or the question after someone already voted, we need to delete the previous results for that question.