웹 기반 움직이는 GIF 제작 서비스

오윤석*, 이정민*
경희대학교 컴퓨터공학과
dhdbstjr98@khu.ac.kr, danmin20@khu.ac.kr

Web-based gif production services

Yunseok Oh*, Jeongmin Lee*
Department of Computer Science and Engineering, KyungHee University

요 약

SNS가 널리 발달한 요즘은 자신의 개성을 표현하는 것이 크게 각광받고 있다. 특히 사진을 단순히 찍어서 SNS에 공유하는 것을 넘어 사진에 글씨를 쓰고 스티커를 붙이는 등 자신만의 커스터마이징을 한다. 본 어플리케이션은 사용자가 업로드한 이미지에 여러 애니메이션을 넣어 움직이는 GIF 이미지를 만들 수 있도록 도와주고, 이것을 쉽게 공유할 수 있는 방법을 제공한다.

1. 서 론

1.1 연구 배경

최근 다양한 스마트 기기의 보급으로 인해 SNS가 널리 사용되고 있다. 특히 요즘에는 페이스북, 인스타그램을 필두로 한 사진 중심 SNS 서비스가 주류를 이루고 있다. 이러한 사진 중심 SNS 서비스가 널리 사용되면서 사용자들은 자신의 사진에 여러가지 효과를 넣기 시작했다. 사진에 텍스트를 추가하고, 스티커를 추가하면서 점점 더 개성 있는 이미지가 만들어졌다.

하지만 정적인 이미지는 사용자의 개성을 표현하는 데에 한계가 있다. 이제는 여러 SNS의 사진 편집 툴도 제공하는 스티커 이미지 종류 정도를 제외하고는 기능이 거의 비슷하다. 점점 다양화되는 사용자의 개성을 표현하기에는 더 이상의 기술 발전이 어려워 보인다.

"웹 기반 움직이는 GIF 제작 서비스"는 정적인 이미지의 한계를 보완하고자 탄생하였다. 본 프로젝트는 정적인 사진만으로는 표현할 수 없는 개성을 움직이는 GIF 이미지로 마음껏 표현할 수 있도록 기능을 제공하고, 이것을 쉽게 공유할 수 있는 방법을 제공한다.

1.2. 연구 목표

사용자가 쉽게 다양한 기능을 사용할 수 있도록 한다는 목적으로 아래와 같은 세 가지 목표를 설정하였다.

첫 번째 목표는 "직관적인 UI 구성"이다. 지금은 본 프로젝트와 유사한 웹 기반 어플리케이션이 없는 상태이다. 그래서 사용자가 처음 보는 서비스의 UI에 쉽게 적응할 수 있도록 하는 것이 무엇보다 중요하다. UI를 다른 상용 프로그램과 유사한 형태로 개발하여 사용자가 쉽게 적응할 수 있도록 유도해야 한다.

두 번째 목표는 "크로스 플랫폼 지원"이다. 웹 기반 서비스의 장점을 살려 크로스 플랫폼 사용이 가능해야 한다. 특히 무엇보다도 모바일 환경에서 PC에 비해 유저 경험이 크게 차이가 나지 않도록(가능하다면 모바일 사용자 경험이 더 좋도록) PC와 모바일 양쪽에서 사용 가

능한 반응형/적응형 웹 서비스로 구성해야 한다.

세 번째 목표는 "단순하지만 다양한 기능 제공"이다. 사용자의 개성을 마음껏 표현할 수 있도록 다양한 기능을 제공해야 한다. 브러시, 텍스트 입력 등 사용자가 어떤 기능을 요구할지를 사전에 잘 파악하여 요구사항에 맞는 다양한 기능을 제공해야 한다. 하지만 각 기능들이 너무 복잡하다면 사용자가 어플리케이션을 이용할 때 피로감을 느낄 수 있다. 따라서 최대한 직관적으로 이해할 수 있는 단순한 기능을 위주로 다양하게 조합할 수 있도록 한다.

2. 관련연구

2.1 개발환경

2.1.1 JavaScript

JavaScript는 HTML과 CSS로 만들어진 웹 페이지를 동적으로 제어할 수 이는 언어이다. 웹 브라우저에서 실행하는 스크립트 언어이며, 가볍고 빠르다는 장점이 있다. HTML 페이지 변경 및 HTML 엘리먼트의 추가나 제거, CSS 및 HTML 엘리먼트의 스타일 변경, 유저의 이벤트에 대한 스크립트 실행 등의 기능을 제공하여 HTML 웹 페이지를 동적으로 만들어 준다. 그 외에서 웹 브라우저 제어를 통한 쿠키/세션 등의 설정과 조회, AJAX 기술을 통한 웹 서버와의 통신 등의 기능도 수행한다.

문법은 es6 이상을 사용하며, Es6의 경우 이전의 문법과 달리 default parameter, arrow function, let/const, template 문자열 등이 추가되어 더욱 효율적인 개발이 가능하다. [1]

2.1.2 TypeScript

TypeScript는 JavaScript에 타입을 부여한 언어로, 정적 타이핑을 지원하므로 컴파일 단계에서 타입 에러를 미리 방지할 수 있다는 장점이 있다. 명시적인 정적 타입 지정은 개발자의 의도를 명확하게 코드로 기술할 수 있으며, 이는 코드의 가독성을 높이고 예측할 수 있게 하며 디버깅을 쉽게 한다. [2]

2.1.3 React.js

React는 웹 어플리케이션의 화면을 개발하게끔 해주는 라이브러리이다. 컴포넌트 기반으로 재사용성이 뛰어나며, 기존의 Real DOM 기반의 웹 어플리케이션과 달리 Virtual DOM 기반으로 돔이 구성/재구성되므로 가볍다는 장점이 있다. 또한 선언적인 프로그래밍을 할 수 있도록 하며, 이러한 선언형 성격에 맞게 컴포넌트 개발에 대해 Jsx 문법을 제공한다. 즉, jsx를 얻기 위한 알고리즘(컴포넌트의 변경사항을 체크하는 알고리즘, 리렌더링 여부에 대한 알고리즘 등)에 대한 구현이 별도로 필요하지 않다. [3]

2.1.4 Next.js

React는 Client Side Rendering으로 이루어지는 single page application 라이브러리로, Next.js는 이러한 React의 Server Side Rendering을 쉽게 구현할 수 있게 도와주는 프레임워크이다. SSR뿐만 아니라 SSG 또한 구현가능하며, CSR, SSR, SSG를 자유롭게 조합하여 개발할 수 있다는 것이 큰 장점이다. 또한 automatic splitting을 지원해주므로 SEO 친화적인 웹 페이지를 개발할 수 있으며, SSR 지원에 따른 추가적인 custom api server를 구축할 수 있다.

또한 에셋에 대한 dynamic import 기능으로 크기가 큰 이미지에 대한 렌더링을 최적화할 수 있다. [4]

2.2. 오픈소스

2.2.1. dhdbstjr98@gif.js

https://github.com/dhdbstjr98/gif.js

기존 gif.js(https://github.com/jnordberg/gif.js)을 개선하여 본 프로젝트를 위해 직접 만든 라이브러리이다. 기존 gif.js는 gif encoding 과정에서 성능 개선을 위해 web worker 기술을 사용하는데, 이 과정에서 별도의 파일이 필요해 webpack 등을 통해 bundle할 때 gif.worker.js라는 별도의 파일을 따로 넣어주어야 하는 문제점이 있었

다. 이 점을 직접 개선하여 npm에 배포하고 본 프로젝트에도 사용하였다.

2.2.2. Toast UI Image Editor

https://ui.toast.com/tui-image-editor

https://github.com/nhn/tui.image-editor

html canvas를 사용한 이미지 에디터로, 이미지를 편집할 수 있는 오픈소스 라이브러리이다.

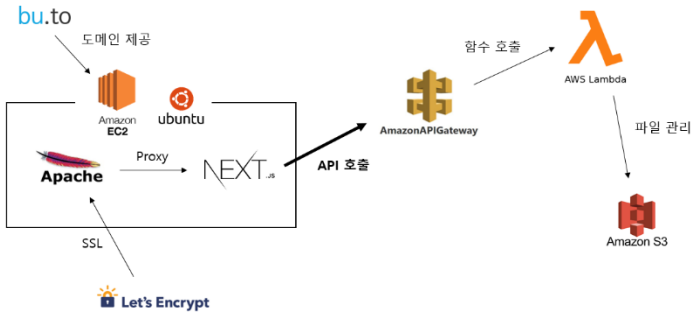
fabric.js(http://fabricjs.com/) 기반으로 구축되어 있다. [6]

2.2.3. Fabric.js

http://fabricjs.com/

html canvas에 대한 다양한 api를 제공하는 라이브러리로, Tui Image Editor 또한 이 라이브러리를 사용하고 있다. [5]

2.3. 배포 환경



2.3.1. Front-end

AWS EC2를 이용해 배포하였다. 프로젝트 팀원이 개별 운영중인 bu.to 서비스에서 도메인을 발급받고, Let's encrypt에서 SSL 인증서를 발급받아 apache2에 적용했다. 이후 next.js를 apache2를 통해 프록시 접속을 하여 배포를 완료했다. https://gif-generator.bu.to를 통해 접속할 수 있다.

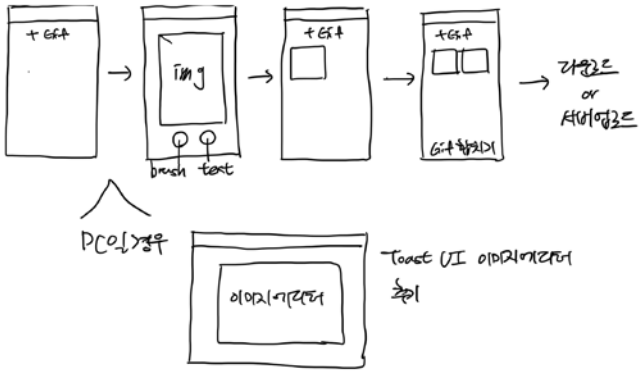
2.3.2. Back-end

AWS S3에 저장 공간을 만들고, AWS Lambda 서비스로 S3와 연동하여 gif 업로드 함수, gif 다운로드 함수를 만들었다. 그리고 이것을 AWS API Gateway를 통해 웹 환경에서 접속할 수 있도록 배포하였다.

3. 프로젝트 내용

3.1 시나리오

3.1.1. 초기 구상 시나리오

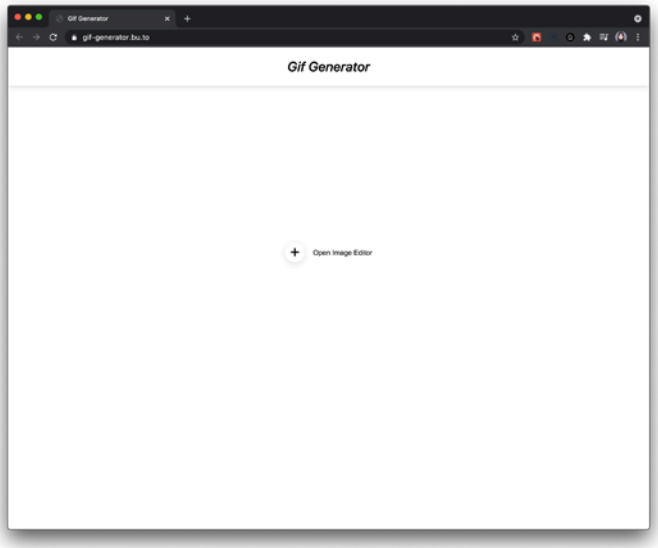


모바일과 PC 모두 똑같은 기능을 제공하기로 계획하였으나, 보다 넓은 PC 화면의 경우 추가적인 기능을 더 제공해도 좋겠다는 아이디어로 이미지 에디팅 기능을 추가해주었다. PC 화면을 먼저 구현하고 추후 미디어쿼리를 사용해 반응형을 구현하고자 했다.

3.1.2. 완성된 시나리오

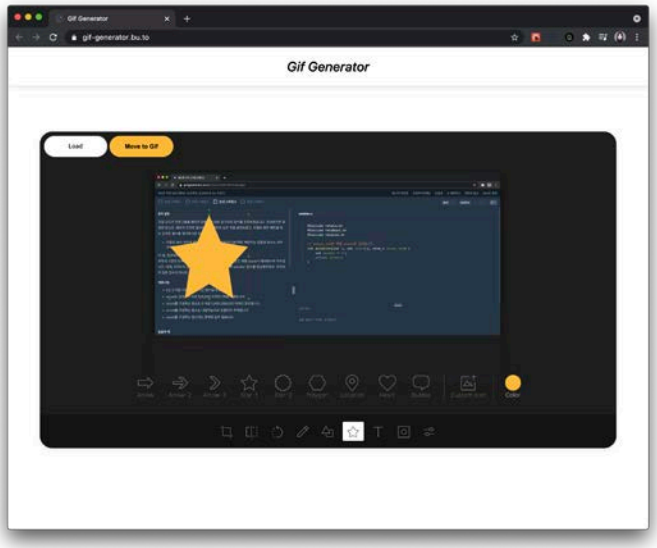
3.1.2.1. 초기화면

사이트에 처음 진입 시, 이미지 에디터를 여는 버튼이 나타나도록 하였다.



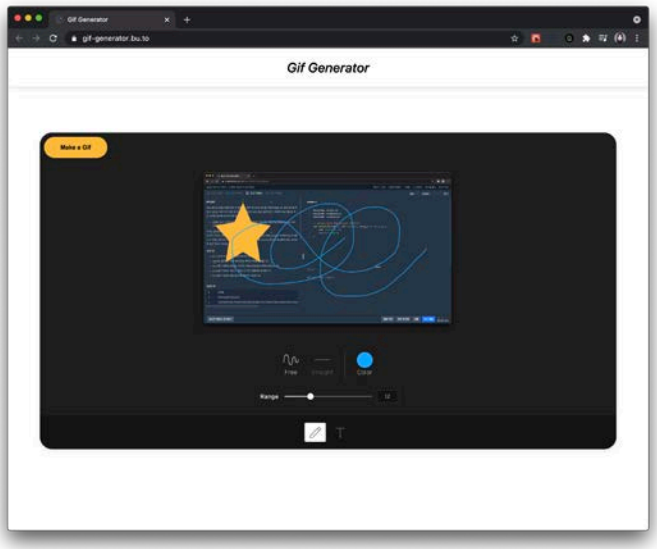
3.1.2.1. 이미지 에디터

이미지 에디터에서는 사진을 로드하여 필터, 크기 조절, 다양한 효과, 도형 그리기 등의 다양한 편집기능을 사용할 수 있다. Move to Gif 버튼을 클릭하면 편집된 이미지가 Gif 에디터로 넘어가게 된다.



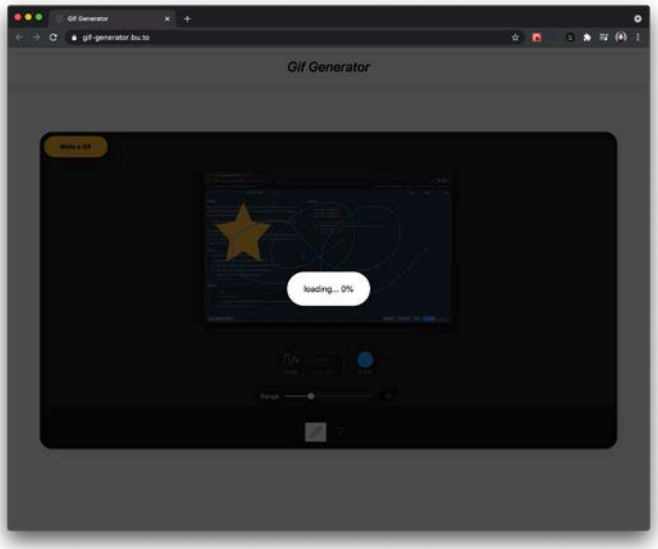
3.1.2.3. Gif 에디터

Gif 에디터에서는 브러시 또는 텍스트를 이용하여 gif를 생성할 수 있다. Make a Gif 버튼을 누르면 gif 생성이 시작된다.



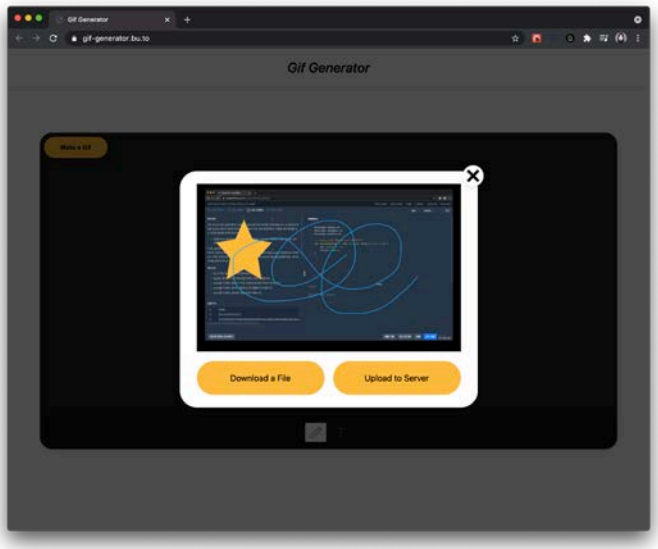
3.1.2.4. Gif 생성

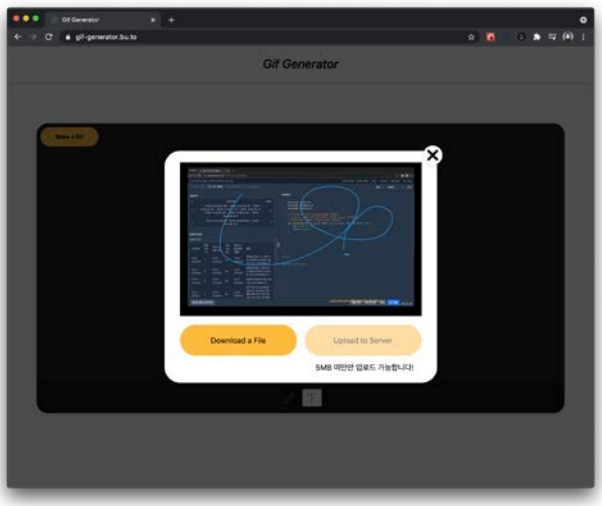
Gif가 생성되는 과정을 퍼센트로 확인할 수 있다.



3.1.2.5. Gif 생성 완료

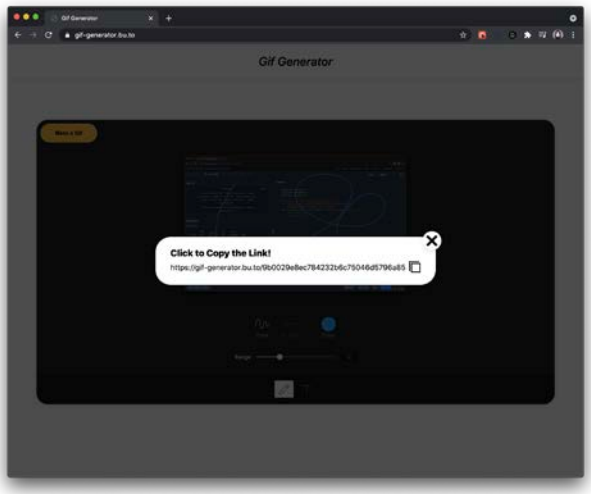
Gif가 생성 완료되면 미리보기 이미지와 함께 파일을 컴퓨터에 다운로드 받을지, 서버에 업로드할지 선택할 수 있는 버튼이 뜨게 된다. 이 때 생성된 gif의 파일 크기가 5MB 이상일 경우에는 서버에 업로드가 불가능하다.

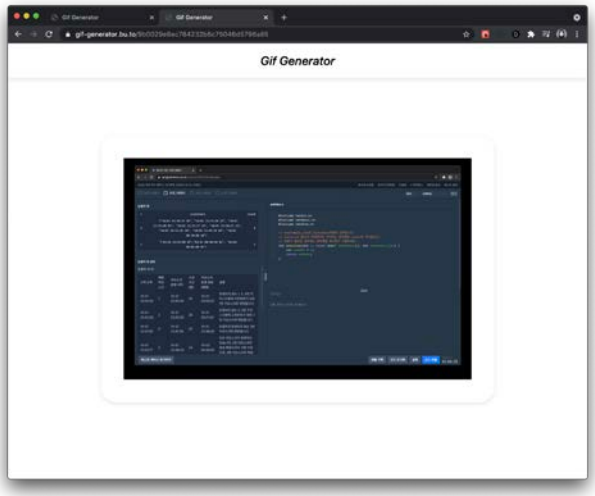




3.1.2.6. 서버 업로드

서버에 업로드할 시, 해당 이미지에 대한 url이 생성된다. 주소를 클릭하면 클립보드에 복사되며, 해당 주소를 통해 해당 이미지를 공유 또는 감상할 수 있다.





3.2. 요구사항

3.2.1. UI 요구사항

웹 기반 어플리케이션이기 때문에 장점을 살려 크로스 플랫폼으로 서비스를 제작하였다. 특히 PC와 모바일 모두에서 사용할 수 있도록 반응형/적응형 웹 서비스로 구현하였다. 특히 기존에 본 프로젝트와 유사한 웹 기반 GIF 제작 도구는 없었기 때문에 사용자가 본 어플리케이션에 쉽게 적응할 수 있도록 통상적으로 자주 사용되는 UI 구성을 최대한 반영하였다.

3.2.2. 서비스 품질 요구사항

사용자가 업로드한 이미지는 GIF 기술이 허용하는 한도 내에서 화질이 크게 훼손되지 않아야 한다. 애니메이션 효과는 다양하지만 복잡하지 않게 한다는 원칙으로 구성한다. 특히 브러시, 텍스트 입력처럼 애니메이션의 시작과 끝이 명확한 것들로 구성하여 사용자가 어플리케이션을 사용할 때 헷갈리지 않도록 직관적인 서비스를 제공할 수 있도록 하였다. 애니메이션 기능 외에도 일반적인 이미지 편집도 제공하여 하나의 이미지 편집을 위해 여러 툴을 사용하는 번거로움을 줄였다.

3.2.3. 안정성 요구사항

사용자가 이미지를 만들고 이것을 서버에 업로드해서 다른 사용자들과 공유하기로 결정하였다면 이 이미지는 안전하게 보관되어야 하며, 이미지가 손상되지 않도록 해야 한다.

4. 결론 및 기대효과

웹사이트의 접근 용이성에서 보았을 때, 이미지 편집을 웹 어플리케이션으로 간편하게 작업함으로써, 쉽고 빠른 이미지 편집에 대한 유저들의 니즈를 충족시켜 줄 수 있을 것으로 기대된다. 또한 기존의 일반적인 이미지 편집기와 달리, 텍스트와 드로잉에 대한 gif를 추가적으로 제작할 수 있게 함으로써 요즘의 인터넷 문화, 다양한 콘텐츠 제작과 소비에 대한 니즈 또한 충족시켜 줄 수 있을

것으로 기대된다.

5. 발전 방향

단순한 UI를 제공하기 위해 애니메이션의 시작 지점과 끝 지점이 명확한 그림 오브젝트인 브러쉬와 텍스트 두 가지만을 이용한 애니메이션 생성만 지원했다. 그림 오브젝트를 제한하지 않고 사용자가 시작 지점과 끝 지점을 직접 설정할 수 있도록 한다면 더 다양한 애니메이션 생성을 할 수 있을 것으로 기대된다.

부가 기능으로 제공한 이미지 업로드 기능은 AWS API Gateway의 한계로 5~10MB의 작은 이미지만 업로드가 가능했다. API Gateway를 사용하지 않고 별도의 웹서버를 구축하는 방법으로 제한을 해제하거나, 원본 이미지와 애니메이션 오브젝트를 따로 전송해서 서버에서 GIF를 만들도록 개선한다면 사용자가 더 편하게 이용할 수 있을 것으로 기대된다.

6. 참고문헌

[1] Javascript 공식 문서
https://developer.mozilla.org/ko/docs/Web/JavaScript
[2] Typescript 공식 문서
https://www.typescriptlang.org/docs/
[3] React 공식 문서
https://ko.reactjs.org/docs/getting-started.html
[4] Next.js 공식 문서
https://nextjs.org/docs/getting-started
[5] Fabric.js 공식 홈페이지
http://fabricjs.com/
[6] Toast UI Image Editor 공식 홈페이지
https://ui.toast.com/tui-image-editor