Dat Ngo Tat

Print in 3D Web Application

Technology and Communication
2018

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

ABSTRACT

Author	Dat Ngo Tat
Title	Print in 3D Web Application
Year	2018
Language	English
Pages	39
Name of Supervisor	Timo Kankaanpää

The customer of the product is Origo Engineering Oy, which is an engineering company that provides mechanical engineering and 3D printing service. The mission is to develop a Web Application for this company that takes 3D printing request of quotation from the customers and sends the request information for both ends.

The focus of the Web Application is to guarantee the security of the order information since 3D files and customer details are valuable, confidential and time-consuming to design. The application also needs to be user-friendly as customers are mostly non-IT-related. Finally, since the web application involves handling sizeable data, its performance speed needs to be considered.

To obtain the mentioned criteria, the technology used are ReactJS for front-end development, Nodejs/PHP for backend and MySQL/MongoDB as database.

The result of the thesis work contains the product design and implementation as well as test cases based on customer requirements. It is tested and performs the required features, which involve securely receiving and sending requests as a Web Application.

CONTENTS

APPENDICES

LIST OF FIGURES AND TABLES

1 INTRODUCTION

- Introduction to 3D Printing Market

Online Ordering is a form of electronic commerce. Using this method, customers can send their requests for purchasing products to the seller and describe their desired features that products should have. /1/

3D printing is defined as a process that includes multiple steps to join a material into a shape based on an electronic sample using 3D printing machine. /2/

3D printing became popular in the 1980s and since then the market has grown exponentially. Moreover, it is expected to still grow in the next ten years, which creates the need to apply Online Ordering to this market. This project can be considered as an example of building an online shopping platform for 3D printing.

- The project aims:

The project develops an online request for a quotation solution for 3D printing services for Origo Engineering Oy. Their customers can login using Google accounts, upload their 3d image files, choose 3D printers, materials, colors, infill and layer height available. The uploaded 3D model is shown in the UI as a preview. Once the requests for quotation are confirmed, they are sent to their email and Origo's email, which enables price and delivery time quotation.

The Web Application is designed using ReactJS, HTML5 and CSS for user interface. For the server side, PHP is chosen.

2 RELEVANT TECHNOLOGY

Below is the introduction of the technologies used in this project.

2.1 Front-end

2.1.1 HTML5

HTML, stands for Hyper Text Markup Language, is the markup language used for writing Web pages. It makes the document well-structured with the structural semantics such as headings and paragraphs using tags, which specify the type of data. It also allows Web apps taking inputs of many types such as file, text and Boolean. Furthermore, HTML convey the structure of the document, the relationship of its content to each other, helps user link the pages together. The possible languages supported by HTML are JavaScript and CSS, which allows improving UI appearance and handle events, mimic data. HTML5 is the version used in this project. /3/

2.1.2 CSS

CSS, stands for Cascading Style Sheets, is a stylesheet language to handle to appearance of markup languages like HTML. CSS has a set of rules to set the identify and set style for document elements. CSS could be applied to HTML as inline directly to an element, style block or external. The external CSS allows multiple web pages adjustment if the pages follow some special rules declared in the CSS file.

2.1.3 ReactJS 16.3

ReactJS is an open source JavaScript library for designing single page user interfaces, which is created by Jordan Walke, a software engineer at Facebook in 2013. The special characteristic of React is turning multiple web-pages application into a single-page application using its components, routers and states. This feature boosts the performance of the application and the developing speed since components could be reused. ReactJS could be written using ECMAScript 6 (ES6),

which lessens the code and makes it more transparent. React is used in the web application of multiple large firms like PayPal, Airbnb and Apple. /4/

2.1.4 Three.JS

Three.js is a cross-browser JavaScript library and Application Programming Interface, created by Ricardo Cabello to GitHub in April 2010. This is used to create and display animated 3D computer graphics in a web browser using WebGL.

2.1.5 Webpack

Webpack is a static module bundler for modern JavaScript application. It combines the modules in one project into a bundle.

Webpack is used to bundle multiple ReactJS file with JSX and ES6 syntax into a bundle file.

2.2 Back-end

2.2.1 PHP

PHP, which stands for Hypertext Preprocessor, was created by Rasmus Lerdorf in 1994 and is a scripting language that can be imbedded into HTML or generate HTML. PHP code is executed by a PHP interpreter by request on the server side and data is sent to the client side. PHP makes pages dynamic, its content change based on conditions and user interaction. /5/

.

2.2.2 NPM

Npm is a package manager for JavaScript, written by Isaac Schlueter, consisting of a command line client and npm registry, an online database for packages. NPM is the default package manager for NodeJS. NPM can install all the project dependencies through the “package.json” file using command line. /7/

In this project, NPM is used to execute the script in package.json to trigger NodeJS code for creating local server side and available ReactJS HTML render.

2.3 3D printing technology

3D printing is the processes of creating a three-dimensional object by joining the material. Objects can be in any shape and is produced from a 3D model by adding material layer by layer.

There are 3 steps in 3D printing:

- Firstly, 3D object is designed into a file using CAD-software.
- Secondly, material is chosen. This step involves choosing material types, color and thickness of the material. After the selection finishes, printing process starts.
- Finally, after the first version of the print. Adjustment is needed to make objects usable by sanded, lacquered or painted.

In this project, the application can handle 2 types of 3d files which are STL and OBJ.

3 PROJECT DESCRIPTION

Origo Engineering Oy needs a solution for their 3D printing customers to place a 3D printing request for a quotation. The solution proposed is a web application where the customer can upload 3D designs, choose available characteristics of their printed object and provide their personal information.

The users use the solution mostly with a PC and a web browser. The users are new or existing 3D printing customers of Origo Engineering Oy and engineers of the company supervising the solution.

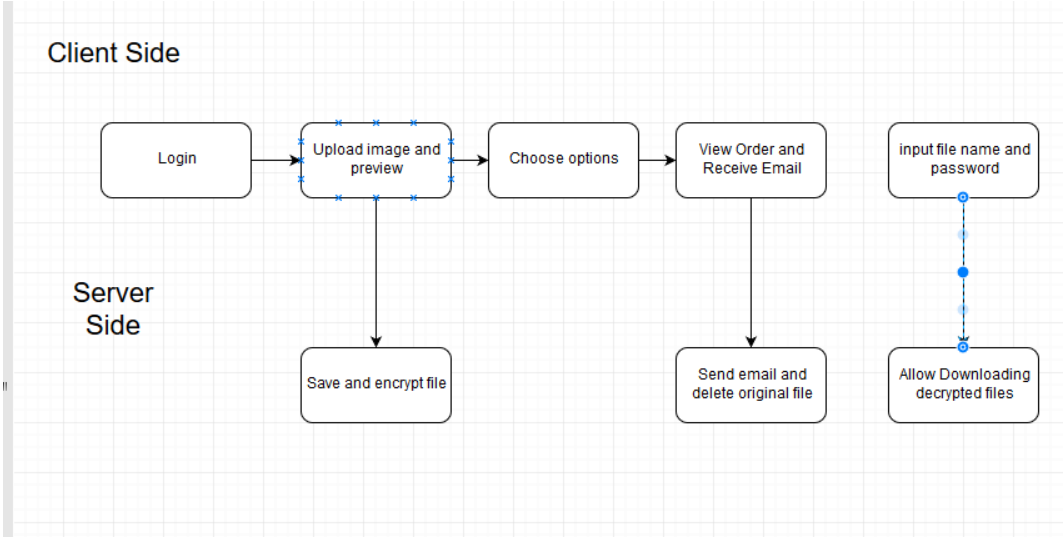


Figure 1. Flow of process steps

3.1 General View

The solution is used to upload 3D drawings and keep the drawings safe and secure in the server while providing the printing service. Once the customer has uploaded the drawing, the data is used to calculate the material need and printing time. The price will initially be calculated by receiving an email and manually using the tools for clarifying the volume and printing time for the request. In the further version, price calculation is automated, and the price is generated without any interaction

with Origo engineers. After receiving the quotation by email, the customer can place an order. The customer can ask for more than one copy for their product.

After printing the 3D object, the Origo engineers make the delivery and billing of the object by their current method.

After customer sends 3D files, they are stored by the server. Server encrypts the received files and stored to the server. A decryption key is given to Origo Engineering Oy to view and deploy 3D files.

3.2 Use cases

Use cases for the project is specified during customer meeting and team discussion during analyzing state. There are 3 priority level for each requirement:

- Priority 1: Functions in this category are compulsory as they are basic function for Origo to provide their service.
- Priority 2: Functions that are not essential but expected in deployment.
- Priority 3: Functions which are suggested to improve the quality of the product.

Below is the use case diagram of every functions:

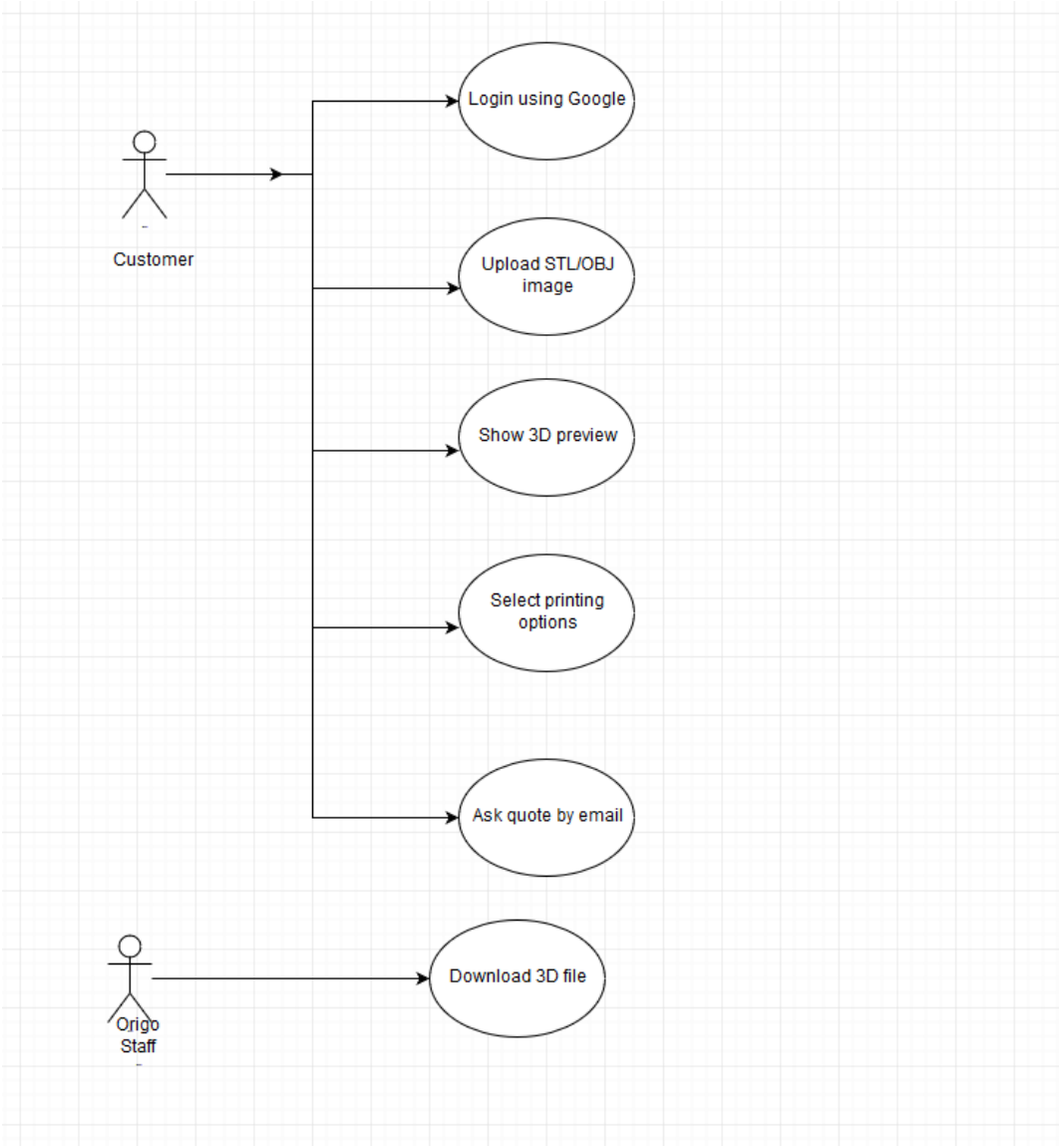


Figure 2. Use-case diagram

There are two types of users of the application: Origo members and their customers. Table 1 shows the priority level specified for each function:

ID	**Description**	**Priority**
F1	Login using Google	1

F2	Upload STL/OBJ image	1
F3	Show 3D preview	2
F4	Select printing options	1
F5	Ask quote by email	2

Table 1. Table of requirements

4 IMPLEMENTATION

This chapter describes the implementation of the techniques in the project

4.1 Client side

4.1.1 Structure

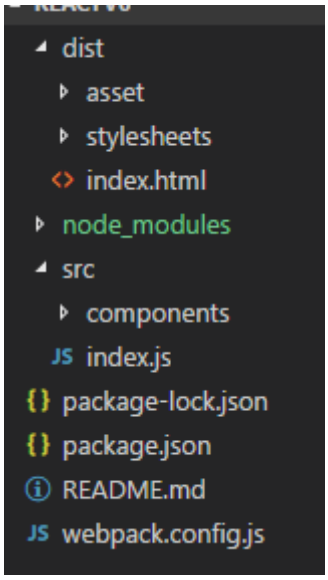


Figure 3. Client-side structure

The folder and files function are explained as below:

- dist: there are bundle.js file, the React compiled file, located in “asset”, css file, located in stylesheets and the main page index.html.
- node_modules: contains all the dependencies needed for the client side.
- Src: there are index.js, the main entry for webpack to build bundle.js and folder components, which contains every needed React component.
- Package.json and package-lock.json: the files for project description.
- Readme file
- Webpack file

4.1.2 Execution

In package.json, all the dependencies are listed. Using Windows Command Prompt, execution can be done as follow: *npm install* (to install the dependencies and *npm start* (to run the application).

If successfully built, the application should be accessed in accordance to the host specified in webpack.config.js using a web browser.

Below is the main page of the application after built:

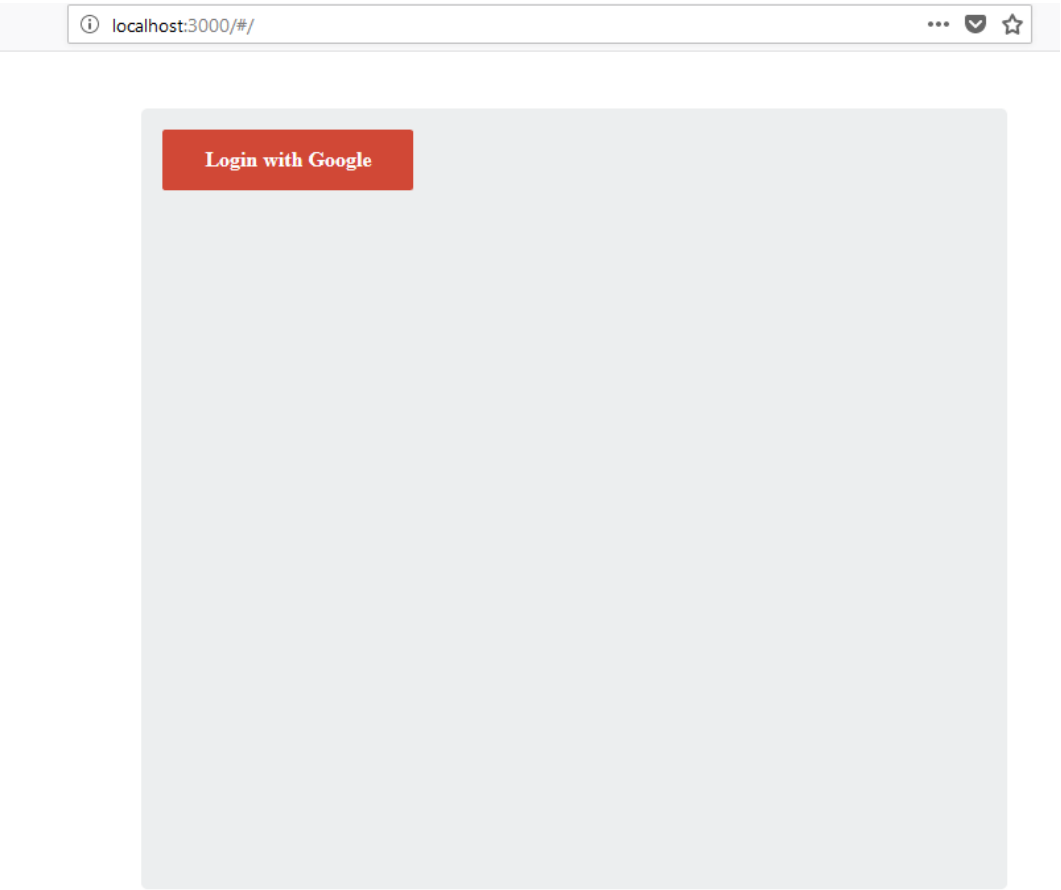


Figure 4. login view

4.1.3 Function explained

- **Routing**: Index.js is the entry point of other React components. All the routes for the application are specified here. The method for connecting

pages in the application is React Router. Routes specification are shown in figure 5.

```
window.React = React

render(
    <Router history={hashHistory}>
        <Route path="/" component={Login}/>
        <Route path="/order" component={App}/>
        <Route path="/completion" component={Completion}/>
    </Router>,
    document.getElementById('react-container')
)
```

Figure 5. Index.js router part

There are three paths available in the application. The main path is for the authentication purpose. Only after successful login, the user can continue to path "/order". Path "/order" shows an upload feature and available options for the 3D object. In the final step of "order", the user receives an order confirmation where he/she sees his/her selections and a button for sending email. After the button pressed, the application shows the path"/completion", which informs that the mail is sent and the user can logout by clicking the logout button. The user is then logged out and application is back to the main path.

Login/Logout: Login button is a component provided by package "react-google-login", which has four props.

Figure 6 shows the login button.

```
{   (this.state.isAuth==false) ?
    <GoogleLogin
        clientId="[redacted].googleusercontent.com"
        buttonText="Login with Google"

        onSuccess={this.successResponseGoogle}
        onFailure={this.errorResponseGoogle}
    />
        : <div>
            <p>You are logged in, click next to continue</p>
            <Link to='/order'>
            <Button className="Next-button" color="success" >Next</Button>
            </Link>
        </div>
}

{this.state.error && <p>{this.state.error}</p>}
</div>
</Jumbotron>
```

Figure 6. Login code

Prop clientID take the client ID provided by google API to enable the Google Login function. If user inputs a valid Google username and password, onSuccess function is activated. In this application, onSuccess is binded to successResponseGoogle function, which store the user identification into session using sessionStorage.setItem() and sessionStorage.getItem() function. onFailure is activated if user is failed to login, which, in this application, will display an error.

Logout button has the function of redirecting to the main path and clear all the session storage.

Below is the logout function in logout button:

```
logout= () => {

    sessionStorage.clear();
    window.location.href='/'
}
```

Figure 7. Logout code

- **3D review and file upload**

Component Printer renders two rows. The first row contains two columns. The first column is a form for uploading 3D drawings. Inputs taken are sent to PHP to be handled on the server side. File MIME types are limited to. slt and obj to avoid error in the 3D display. The MIME types restriction is also handled at server side by PHP. The second column displays 3D preview based on what the users upload to the sever side. The component used to display 3D drawings is Preview3d, which uses THREE.JS to display 3D objects. Each time users uploads the file, the page is reloaded. Hence, setting file name as a state is not possible since states are reset as pages reload. Due to this characteristic of React component, file name is set to the session storage. Preview3D receives the state as a prop and displays the file. Since it is rendered after each time the page is loaded, functions to handle the display are imbedded into function ComponentDidMount of Preview3d. This is one of a function in life-cycle function of React that is auto-triggered every time the pages are loaded, and components are mounted.

The second row of component Printer contains two HTML divs which have the information of two available printers at Origo Engineering Oy. There are images of the printers that show the printers ‘image. Printers descriptions are under each image, that specify the volume, printing accuracy and materials available for the respective printer.

Figure 8 is the printer choosing pages.

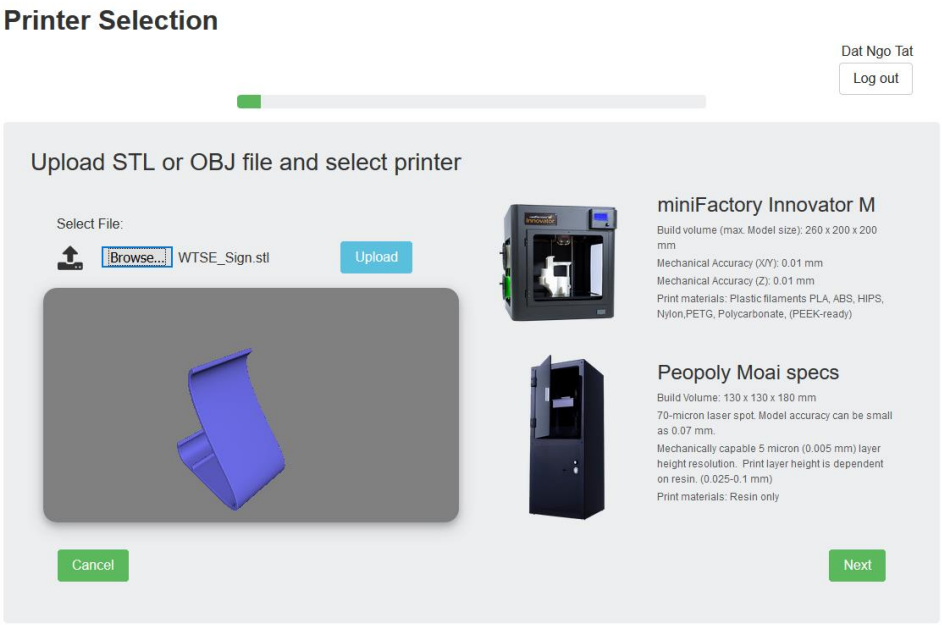


Figure 8. Printer page display

- **Single page feature**

Path "/order" of the application contains 4 pages: choose printers form, colour and material table, infill and thickness height form and personal information. All the components that needed to make the pages are children components of a component named App. Depend on the state of prop "step", App shows the page.

Below shows how prop "step" is set and which components will be rendered in each step.

```
{(this.state.step==1) ?
<div>
    <Header step={this.state.step} handleHomePage={this.handleHomePage}/>
    <Printers
        isAuth={this.isAuth}
        data={data}
        selectedPrinter={this.state.selectedPrinter}
        choosePrinter={this.choosePrinter}
        handleNextPage={this.handleNextPage}
        handleHomePage={this.handleHomePage}
    />
</div>
:   (this.state.step==2) ?
        <div>
            <Header step={this.state.step} handleHomePage={this.handleHomePage}/>
            <ColorTable
                isAuth={this.isAuth}
                materials={data.options[this.state.selectedPrinter-1].steps[0]}
                addMaterialColor={this.addMaterialColor}
                handleNextPage={this.handleNextPage}
                handlePreviousPage={this.handlePreviousPage}
            />
        </div>
        : (this.state.step==3) ?
            <div>
```

Figure 9. Render component code

In step one, the App renders two components which are Header and Printers for choosing Printer and upload 3D Drawing.

- **Two-way data binding**

React offers a technique to pass data which is called two-way data binding. To be specific, React components can receive data as state through their props when it is called in parent components. A child component can pass states to its props directly through functions and the when imported and called in parent components, the mentioned states can be arguments of a method in other components or class that calls them. By this mechanism of transferring data, the need for server side to pass data through pages can be omitted.

Figure 10 is an example of two-way data binding.

```
<Header step={this.state.step} handleHomePage={this.handleHomePage}/>
<ColorTable
    isAuth={this.isAuth}
    materials={data.options[this.state.selectedPrinter-1].steps[0]}
    addMaterialColor={this.addMaterialColor}
    handleNextPage={this.handleNextPage}
    handlePreviousPage={this.handlePreviousPage}
    currentState={this.state.colorMaterial}
/>
</div>
```

Figure 10. Two-way data binding

In component App, ColorTable receives the state isAuth of App component as its prop IsAuth. Only if the user logins successfully does isAuth return *true*. Color table only shows the table to choose color and material if its isAuth props received the value *true*. ColorTable ‘s prop material receives a part of an array in 3d.json file (the file contains all the data for rendering in the application, will be explained later in this chapter). Prop materials is store the data about colors and materials displayed based on printer chosen. Prop handleNextPage and handlePreviousPage do not receive data but instead transfer data about the step so that App knows which component to render. Those props are set by 2 buttons “Next” and “Previous”, which have their OnClick function passing the new state of “step”.

Below is the OnClick function of button “Next”:

```
handleNextPage = () => {
    if(this.props.isAuth() && this.props.currentState.color && this.props.currentState.material) {
        this.props.handleNextPage()
    }
    else {
        //Do nothing
    }
}
```

Figure 11. OnClick function of button “Next”

As shown in the code, the prop handleNextPage of the ColorTable is binded to the prop handleNextPage of the App. Next, addMaterialColor is the props that contains

data about the data of the new color and material picked by the user. It is set to the new state of the array “materialist”, which is a state to contain data about the picked option in ColorTable, by the function updateMaterialList in ColorTable, which is shown in the code in figure 12.

```
updateMaterialList = (update) => {
    this.setState(() => ({materialList: {
        color: update.color,
        material: update.material
    }}))
    this.props.addMaterialColor(update);
}
```

Figure 12. Update color and material

After receiving and passing data with ColorTable, prop addMaterialColor is sent to function addMaterialColor as an argument for “add” as shown in figure 13.

```
addMaterialColor = (add) => {

    console.log("addMaterial works", add.color)
    this.setState(() =>
        ({
            colorMaterial: {
                color: add.color,
                material: add.material
            }
        }))
    console.log(this.state);
}
```

Figure 13. Update color and material from Row

colorMaterial is a state in App, which is sent to OrderConfirm when the props “step” is 5, which is a component that shows all the chosen options by user for sending email.

Not only colorMaterial, OrderConfirm receives the whole state of App to display data to user. Here is the state of component App:

```
state = {
    step: 1,
    // take order
    //file, printer, color, material, thickness, infill, customerName, Email, Phone, Amount
    file: "",
    colorMaterial: {
        color: undefined,
        material: undefined,
    },
    thicknessInfill : {
        thickness: undefined,
        infill: undefined,
    },
    cusInfo: {
        customerName: undefined,
        email: undefined,
        phone: undefined,
        amount: undefined,
    },
    selectedPrinter: undefined
}
```

Figure 14. Update color and material to App

File and selectedPrinter is obtained from component Printer, thicknessInfill is obtained from Details, cusInfo is obtained from UserData by the same mechanism, two-way data binding.

In general, with several two-way data bindings, data is transferred among parent components and its child components for displaying data and passing data.

- **Table display**

In this application, table display is done by mapping, which reduces the code length and improve the implementing speed.

The component ColorTable has 2 React child components: TableHeader and TableRow. Each Table Header is a cell in the header part of the table and each TableRow represents a row in the body parts. Mapping helps rendering multiple

component “TableHeader” and “TableRow” without rewriting code. Figure 15 is the code for building the table:

```
<Jumbotron>
    <h3> Color & Material </h3>
    <div className="data-holder">
    <form  onSubmit={this.updateMaterialList} className="add-day-form">
<table id="ctable">
    <thead>
        <tr>
            <th></th>
        {this.props.materials.materials.map((day, index) =>
            <TableHeader key={index} id={day.value} name={day.name} imgSrc={day.img} description={day.description}/>
        )}
        </tr>
    </thead>
    <tbody>

        {this.props.materials.materials[0].colors.map((day, i) =>
            <TableRow key={i}
                color = {day.name}
                colorNumber = {i}
                material = {this.props.materials}
                sendData = {this.updateMaterialList}
                handler ={this.handler}/>
        )}
    </tbody>

</table>
<br/>
</form>
```

Figure 15. Building table

As explained, the prop named materials takes data from the file named “3d.json” when called in component App. The number of headers is the number of materials and the number of rows is the number of colours available from the chosen printer, according to the 3d.json file. Mapping loops through all the objects in the specified branch of an array and give a key for each object, then calls the component as a task for each time it loops through an object. Figure 16 is the appearance of page ColorTable:

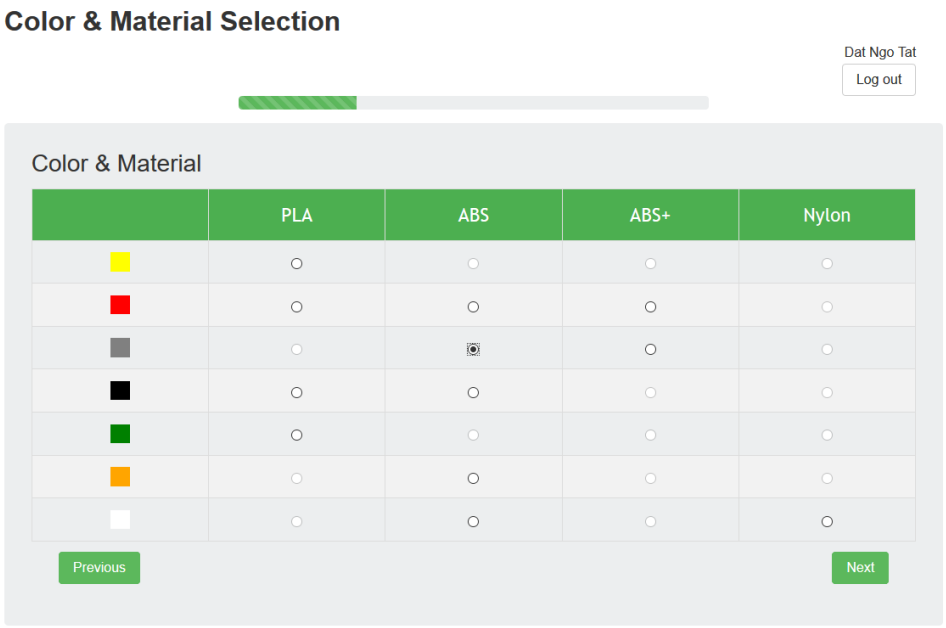


Figure 16. Color & Material display

- **Displaying by data of a json file**

The component App of this application displays data following a rule decided in a file name 3d.json. The json file is created based on the information given by Origo Engineering Oy. By using this method, developers and Origo engineer can modify data as they wish by modifying the json file. Usage could be achieved by importing the json file to React component. After importing, data can be sent and adjust like a state in the component. Passing state is described previously in the chapter. Here is a part of this file:

Figure 17. File 3d.json

For example, in figure 17, by using this, the application knows which a specified printer, color and material table is must display only the colors and materials available for the printer.

4.2 Server side

The server side of the application uses PHP, for testing purposes in local host, WAMP server is used which includes PHP version 5 and 7, Apache 2.4 and MySQL. By default, the root directory is the folder www inside WAMP.

4.2.1 Rendering client side

The main page of the application is index.php. ReactJS is bundled into bundle.js. To render bundle.js, its host needs specifying in index.php. Figure 18 is the content of index.php:

```
<div id="react-container"></div>
<script type="text/javascript" src="http://localhost:3000/assets/bundle.js"></script>
<script type="text/javascript" src="http://unpkg.com/react-google-login@3.1.0"></script>
```

Figure 18. Render method

When this application is tested for making the thesis, bundle.js is hosted on port 3000 and stored under assets directory in the client side by webpack. The online CDN for react google login is needed to enable login feature.

In the final version for customer uses, the bundle.js is bundled by command npm run build, which is a command that is imbedded by the command to execute webpack, which is

```
./node_modules/.bin/webpack-dev-server
```

This command calls the webpack server that is located inside folder node_modules under hidden folder bin. The result of this command is a bundle.js file, which is moved into the file contains php files on the server. This allows rendering React component without running a second host. However, changes in the user interface in this situation must be done by changing the code inside bundle.js, which requires much more extra time since the file is longer since everything is combined to one file. In this project, the bundle.js file has a length of approximately 90000 lines. One more reason that this method should not be used in the developing process of the project is that language used for making React Components is JavaScript with syntax ES6, but most browsers can only interpret ES 5 syntax and below. Webpack does the works of transpiring the syntax for developers so that in the bundle.js file, all the code are rewritten by Webpack using ES 5 syntax.

In index.js, all components are specified to be inside a div with id equals to “react-container”. Hence, the exact div need declaring in index.php.

4.2.2 File Uploading

In the choose-printer page, a file is uploaded to the server. Action in the form is set to be upload.php. Hence, the file upload.php is created in the root directory. In the form, there are hidden fields that contains information about the google name of users and their googleID, which is a unique string used to distinguish the users. Depending on the googleID, a new folder is created to store the files uploaded by the exact user accounts. For example, the file uploaded by a user with googleID "12345" is named "12345_uploads". By using different folders for each user, the conflicts of more than one user using the service at a same time is solved since in the client side, the url for the 3D preview image is made based on the google ID used at the time. It is not possible for a user to view data of other users without knowing the URL. Later, improvements can be made to strengthen the protection such as setting permission for each file in a certain upload folder. However, this method requires database implementation to manage the login systems. Using $_FILES, all information about the file can be obtained. Through that, the file name and the file type are known by

```
$_FILES['filename']['name']
```

In this case, the upload field in html has the name which is "name". The next step is checking if file type is slt or not by checking the file type with

```
$file_type = $_FILES['filename']['type'];
```

If the file type is not slt or obj, alert pops up to notify users and nothing is uploaded. To avoid duplication when uploading, a file rename is needed. File rename is done by adding the sender name and Google ID and a random number among 1000 and 100,000. The random number is needed if users upload more than one file with the same name since otherwise the old file is overlapped with the new one with the same name. The copy function is used to maintain the one with the original name for the display purpose.

When uploading, the file is stored in a temporary folder which is specified in a file called php.ini in WAMP server. If no action is taken to handle the file, it is deleted. Thus, after valid the file type, it is moved to a folder by function move_uploaded_file. Below is the code for handling uploaded file:

```
if(isset($_POST['upload']))
{
    print_r($_FILES) ;

$file = rand(1000,100000)."-".$_FILES['filename']['name'];
   $file_loc = $_FILES['filename']['tmp_name'];

$file_type = $_FILES['filename']['type'];
if (!file_exists($_POST['senderID'].'_uploads')) {
   mkdir($_POST['senderID'].'_uploads', 0777, true);
}
$folder=$_POST['senderID'].'_uploads/'.$_FILES['filename']['name'];
 //$folder=$folder.$_POST["senderName"]."-".$file;


move_uploaded_file($file_loc,$folder);
copy($folder,$_POST['senderID'].'_uploads/'.$_POST["senderName"]."-".$_POST["senderID"]."-".$file);
```

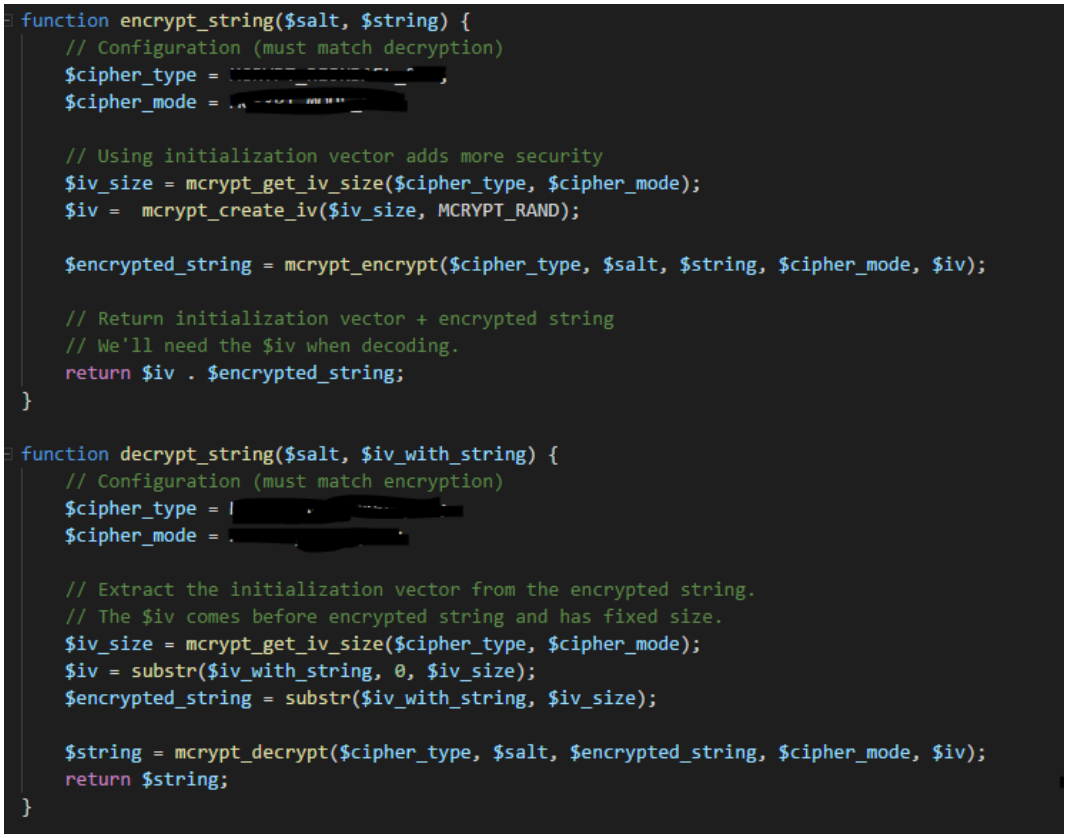
Figure 19. Upload method

4.2.3 File Encryption

After file is uploaded to the server, it is encrypted. However, the unencrypted file is needed to display the 3D image. Thus, the encrypted file is stored in a folder called encrypt_files. After file is moved to uploads file in the server, encrypt function is called. Encrypt function read the file by fopen function, encrypt the string obtained and write it to the new file with the same name as the uploaded file but stored in folder encrypt_files.

The encryption method is encrypting the string in the uploaded file using mcrypt of php. Mcrypt is a replacement in PHP for the Unix crypt command. The algorithms used is AES, stands for Advanced Encryption Standard, a method established by the US National Institute of Standards and Technology in 2001. /9/

Here is how string is encrypted / decrypted in the application:



```
function encrypt_string($salt, $string) {
    // Configuration (must match decryption)
    $cipher_type = [redacted],
    $cipher_mode = [redacted]

    // Using initialization vector adds more security
    $iv_size = mcrypt_get_iv_size($cipher_type, $cipher_mode);
    $iv =  mcrypt_create_iv($iv_size, MCRYPT_RAND);

    $encrypted_string = mcrypt_encrypt($cipher_type, $salt, $string, $cipher_mode, $iv);

    // Return initialization vector + encrypted string
    // We'll need the $iv when decoding.
    return $iv . $encrypted_string;
}

function decrypt_string($salt, $iv_with_string) {
    // Configuration (must match encryption)
    $cipher_type = [redacted]
    $cipher_mode = [redacted]

    // Extract the initialization vector from the encrypted string.
    // The $iv comes before encrypted string and has fixed size.
    $iv_size = mcrypt_get_iv_size($cipher_type, $cipher_mode);
    $iv = substr($iv_with_string, 0, $iv_size);
    $encrypted_string = substr($iv_with_string, $iv_size);

    $string = mcrypt_decrypt($cipher_type, $salt, $encrypted_string, $cipher_mode, $iv);
    return $string;
}
```

Figure 20. Encrypt method

There are more than 30 cipher types and five cipher modes that are available for use.

By specifying the cipher type and mode, an initialization vector (IV) is created. For encrypting, cipher type, mode, string, iv and salt must be provided as arguments. Salt is a random string created by the developer. This is an also requirement for decrypting. The random string is recommended to have the minimum length of the one used in this project to avoid attackers figuring it out.

When the user session is over, the uploads folder is empty. Figure 21 shows the code for empty the folders:

```php
<?php
//before sending email, uploads folder is empty
$files = glob('uploads/*'); // get all file names
foreach($files as $file){ // iterate files
  if(is_file($file))
    unlink($file); // delete file
}
```

Figure 21. Delete file method

There is a separated page in the application that is only used by Origo engineers for decrypting files. The page required the exact file name that store in folder "encrypt_file" in the server and the password string used to encrypt. After inputting correct file name and password, a pop up is shown for downloading the decrypted file, which is the same as the original file sent be customers. This file is fully functional and can be examined by 3d viewing programs.

4.2.4 Mail Sending

This application uses PHPmailer to send mail. PHPMailer is a code library to send email safely that can be downloaded and composed by Composer. Composer is a tool for dependency management in PHP that allows user to install and update the dependencies. /10,11/

Simple Mail Transfer Protocol, or SMTP, is an internet standard for email transmission /12/

SMTP is required by PHPmailer. SMTP can be configured by modifying php.ini file. However, this application use SMTP host of Gmail as an alternative. Below is the code to set SMTP:

```
$mail->isSMTP();


//Enable SMTP debugging
// 0 = off (for production use)
// 1 = client messages
// 2 = client and server messages
$mail->SMTPDebug = 2;

//Set the hostname of the mail serve
$mail->Host = 'smtp.gmail.com';
// use
// $mail->Host = gethostbyname('smtp
// if your network does not support

//Set the SMTP port number - 587 for
$mail->Port = 587;

//Set the encryption system to use -
$mail->SMTPSecure = 'tls';

//Whether to use SMTP authentication
$mail->SMTPAuth = true;
```

Figure 22. send mail method

Mail is taken as a string variable in a hidden form in the html form. The string is a html page with CSS that has React props turn into string as data. Figure 22 shows the form mail that can be received when using the service.

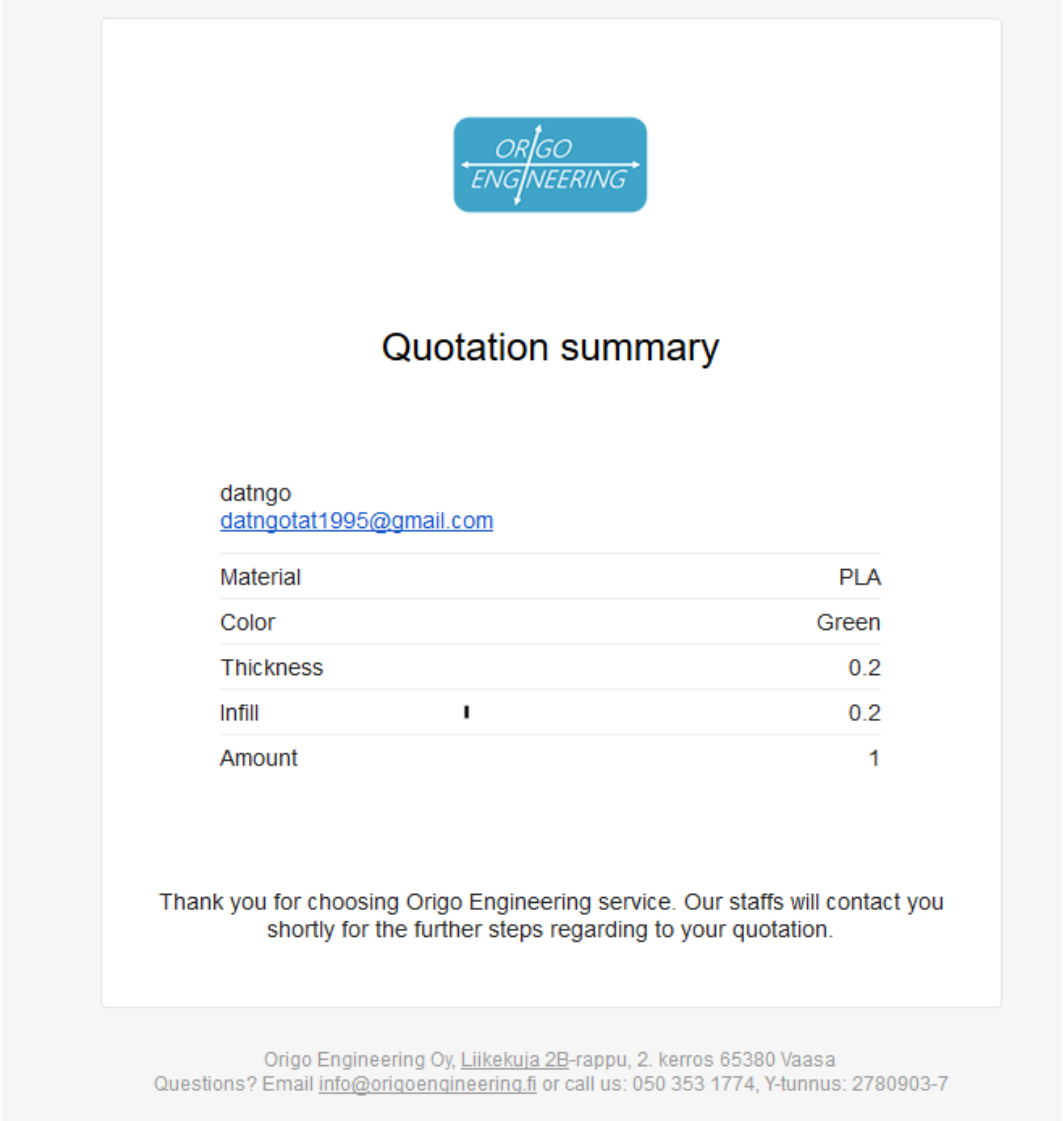


Figure 23. Mail form

5 TESTING AND RESULTS

In this section, test cases are checked to see what is fulfilled and what needs to be improved. As tested, all the pages can be displayed in a dynamic website with no delay in desktop browsers such as Firefox, Microsoft Edge and Google Chrome along with Safari in mobile device. The mail function is tested with Gmail. On Yahoo and Outlook, special characters are displayed incorrectly. There are minor mistakes in the interface such as: only a small part of the Next button is clickable, or the phone number filter only allows a number string of exact 10 numbers.

5.1 Display 3D

The goal of this test is to check whether customer can preview their 3D image after uploading. The image shows the quotation sum up when all the selections have been made.

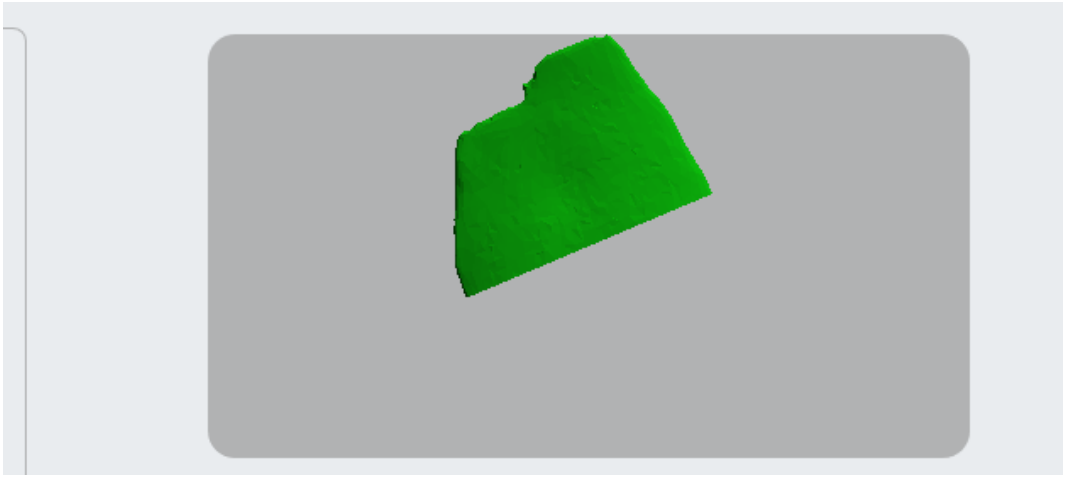


Figure 24. 3D preview

This is the image that is uploaded to the server and the color chosen is green. This preview successfully allows users to zoom and spin the image to check its dimensions.

The limitation is, there are some cases when the uploaded image is not shown in the preview until users spin the frame.

5.2 Login System

This section tests the login feature. Without logging in, if users try to access further pages, they are auto-redirected to the login page thus login is necessary to use the service.

Google ID and Google name of users are successfully taken to distinguish users, which helps Origo engineers for further discussion with customers. After logging out, users can no longer see other pages unless they login again.

However, when logging out outside of the application (for example by using the main page of Google), the customer remains logging in in the 3d printing website until they close the application.

5.3 Upload and Encrypt

Whenever users upload images, files should be save in their own folder based on Google ID and at the same time, encrypted files are created under a separate folder. Below are the folders contain files from one user and the images saved at the same time.

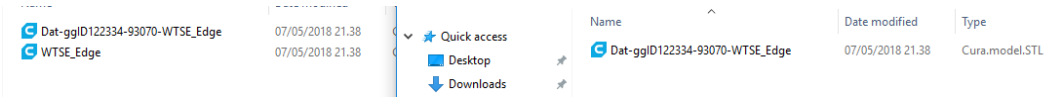


Figure 25. Folder comparation

To prove encryption works, texts of 2 files before and after encryption are shown below. The original one can be read both by human and 3d display software

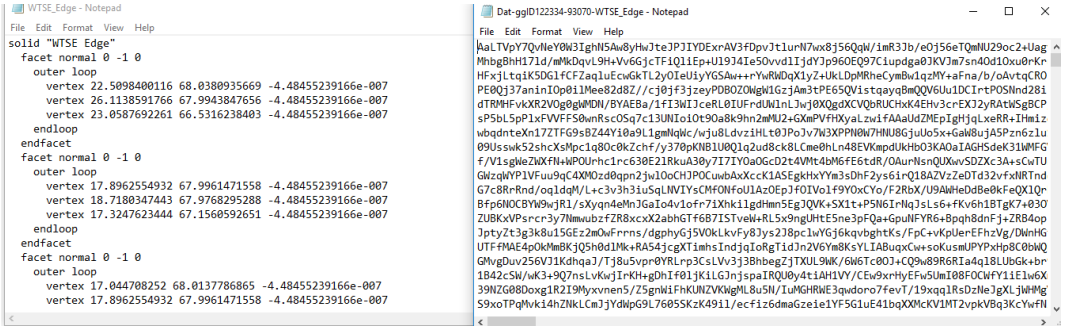


Figure 26. File comparation

6 CONCLUSION

To summarize, the project which is described by this thesis is a web application that provide a solution for a company named Origo Engineering Oy. The solution helps them track the customer quotations and take the information for further contact. It allows uploading and reviewing files as well as choosing selections for the products while encrypt the files to protect them from attackers.

The project is done following strict phases. At the beginning, end and among versions, there are meetings with customer to decide the feature of the web application based on their requirements. The developer translates their wishes into graphs and diagrams for easier design and implementation. There are some difficulties at initial time of the project when new technologies are applied but after the that, everything matches the requirements of customers for each version. The current version is ready for deployment, even though there are new features that can be implemented such as price calculation and user management system.

The thesis can be a model for other 3d printing web application in the future since the demand for this service is higher recently.

References

/1/ Online Shopping:

https://en.wikipedia.org/wiki/Online_shopping

/2/ 3d printing:

https://en.wikipedia.org/wiki/3D_printing

https://www.sculpteo.com/en/3d-printing/3d-printing-technologies/

/3/ HTML

https://en.wikipedia.org/wiki/HTML

/4/ ReactJS

https://en.wikipedia.org/wiki/React_(JavaScript_library)

/5/ PHP

http://php.net/manual/en/intro-whatis.php

/6/ Apache

https://en.wikipedia.org/wiki/Apache_HTTP_Server

/7/ NPM

https://en.wikipedia.org/wiki/Npm_(software)

/8/ Webpack

https://webpack.js.org/concepts/

/9/ Mcrypt

https://en.wikipedia.org/wiki/Mcrypt

/10/ PHPmailer

https://github.com/PHPMailer/PHPMailer/wiki

/11/ Composer

https://getcomposer.org/doc/00-intro.md

/12/ SMTP

https://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol

/13/ Three.js

https://en.wikipedia.org/wiki/Three.js