# **Create a Tax Invoice Generator Using HTML, CSS, and JavaScript**

## **Table of Contents**

1. Introduction
2. HTML Code
3. CSS Code
4. JavaScript Code
5. Preview

**Source Code**

**Step 1 (HTML Code):**

we will first need to create a basic HTML file. In this file, we will include the main structure for invoice generator.

After creating the file make sure to save your HTML document with a .html extension, so that it can be properly viewed in a web browser.

The document begins with the standard HTML structure, including the <!DOCTYPE html> declaration and the opening and closing <html> tags.

The <head> section contains metadata and references to external resources. It includes:

* The title of the document, which is set to "Tax Invoice Generator."
* The character encoding is set to UTF-8.
* The viewport is defined to adjust the width of the page to the device's screen.
* Two links to external stylesheets. One is for the "Material Icons" font, and the other is for the "Roboto" font. These fonts will be used in the document.
* A link to an external CSS file named "styles.css" for additional styling of the document.

The <body> section contains the content of the invoice generator. It consists of company name along with anchor tag i.e <a></a> by clicking this it will open new tab which displays company portal and it also consists of a <div> element with the class "invoice-container" that encloses the entire invoice.

Below the "invoice-header" is the "invoice-body" <div> which contains a <table> element representing the invoice items.

The table has a header row (<thead>) with column headers for "Product Name" "Unit" "Price" "Amount" and "Action."

The table body (<tbody>) initially contains a single row with input fields for product name, unit, price, amount, and a delete icon.

Below the first row, there is another row with dashed cells and an "add" icon. This row serves as a placeholder for adding more rows dynamically.

At the end of the "invoice-body," there is a <div> element with the ID "sum" that contains an input field for the total amount of the invoice. The input field is initially disabled.

Finally, there is a reference to an external JavaScript file named "script.js" at the bottom of the document. This file likely contains additional functionality for the dynamic behavior of the invoice generator.

**index.html**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<title>Dynamic Invoice Generator</title>**

**<meta charset="UTF-8" />**

**<meta name="viewport" content="width=device-width" />**

**<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">**

**<link href="https://fonts.googleapis.com/css?family=Roboto:100,100i,300,300i,400&display=swap" rel="stylesheet">**

**<link rel="stylesheet" href="styles.css" />**

**</head>**

**<body>**

**<div class="invoice-container">**

**<a href="https://www.goraitech.com/" target="\_blank" style="text-decoration: none;"><h1 style="color: plum;">Gorai Technology Solutions</h1></a>**

**<br>**

**<div class="container1">**

**<div id="left">✆ +91-9739403914</div>**

**<div id="center">✉ hr.gts@ggtech.co.in</div>**

**<div id="right">📍4th Main, 21, 6th Cross, Roopena <br>Agrahara, Bommanahalli, Bengaluru</div>**

**</div>**

**<br>**

**<h4>Bill To: </h4>**

**<br>**

**<h5>A2T Technologies</h5>**

**<p>3 T3 Shivaganga Opal Owner Community 17th A Cross Road Bangalore</p>**

**<p>Contact No: <strong>8884266322</strong></p>**

**<p>GSTIN Number: <strong>29AAYCA6958N1ZC</strong></p>**

**<p>State: <strong>29 - Karnataka</strong></p>**

**<div class="invoice-header">**

**<div class="title-date">**

**<h2 class="title" style="color: #ff057e;">TAX INVOICE</h2>**

**<input type="date" placeholder="Date">**

**</div>**

**<div class="space"></div>**

**<input type="invoice-number" placeholder="Invoice No" name="Invoice No" class="Invoice No" id="Invoice No" onkeyup="getInput()">**

**</div>**

**<div class="invoice-body">**

**<table>**

**<thead>**

**<th style="padding-left:12px;">Product Name</th>**

**<th>Unit</th>**

**<th>Price</th>**

**<th>Amount</th>**

**<th style="text-align: right;">ACTION</th>**

**</thead>**

**<tbody id="table-body">**

**<tr class="single-row">**

**<td><input type="text" placeholder="Product name" class="product left"></td>**

**<td><input type="number" placeholder="0" name="unit" class="unit" id="unit" onkeyup="getInput()"></td>**

**<td><input type="number" placeholder="0" name="price" class="price" id="price" onkeyup="getInput()"></td>**

**<td><input type="number" placeholder="0" name="amount" class="amount" id="amount" disabled></td>**

**<td style="text-align: right;"><span class="material-icons">delete\_outline</span></td>**

**</tr>**

**<tr style="padding-left: 20px">**

**<td class="dashed "><div class="float">**

**<a href="#" class="float" id="add-row">**

**<span class="material-icons plus">add</span>**

**</a>**

**</div>**

**</td>**

**<td class="dashed"></td>**

**<td class="dashed"></td>**

**<td class="dashed"></td>**

**<td class="dashed"></td>**

**</tr>**

**</tbody>**

**</table>**

**<div id="sum"><input type="text" placeholder="0.00" name="total" class="total" id="total" disabled></div>**

**</div>**

**</div>**

**<script src="script.js"></script>**

**</body>**

**</html>**

**Step 2 (CSS Code):**

Once the basic HTML structure of the invoice generator is in place, the next step is to add styling to the invoice generator using CSS. CSS allows us to control the visual appearance of the website, including things like layout, color, and typography.

Next, we will create our CSS file. In this file, we will use some basic CSS rules to create our invoice generator.

Here's an explanation of each section:

The first block \* sets the margin and padding of all elements on the page to 0, effectively removing any default spacing.

The body selector styles the body element. It sets the font family to "roboto" and the background color to white.

The .material-icons class is used to style elements with the "material-icons" class. It sets the cursor to a pointer, indicating interactivity.

The .invoice-container class is applied to a container element for an invoice. It centers the container horizontally (margin: auto) and adds padding to the top and bottom (padding: 0px 20px).

The .invoice-header class is applied to the header section of the invoice. It uses flexbox to display its contents in a row (display: flex), adds padding to the top and bottom (padding: 70px 0%), and sets the width to 100%.

The .title, .date, and .invoice-number classes style various elements within the header section. They set font sizes, letter spacing, and colors for text.

The .space class is applied to a container that occupies 50% of the width.

The table selector styles table elements. It sets the table layout to "auto" and the width to 100%.

The th and td selectors style table headers and table cells, respectively. They set padding, borders, font sizes, letter spacing, and colors for text.

The .dashed class adds a dashed border-bottom style to elements.

The .total class styles elements represent the total amount. It sets the font weight, font size, and color.

The input[type=number] and input[type=text] selectors style input elements of type "number" and "text," respectively. They set text alignment, max width, font size, padding, border, and outline properties.

The input[type=text]:focus and input[type=number]:focus selectors style the focused state of the input elements. They add a border radius and box shadow to create a visual effect.

The input::-webkit-outer-spin-button and input::-webkit-inner-spin-button selectors style the appearance of spin buttons for number inputs in webkit-based browsers.

The input[type=number] selector overrides the appearance of number inputs in Firefox.

The .float class styles a floating circular element. It sets the width, height, background color, text color, border radius, text alignment, and box shadow.

The .float:hover class styles the floating circular element when hovered. It changes the background color.

The .plus class adds a margin to the top of an element.

The #sum selector styles an element with the "sum" ID. It sets text alignment to the right and width to 100%.

The #sum input[type=text] selector styles text inputs within the element with the "sum" ID. It sets the width to 100% and overrides the font size and text color.

The @media rule applies styles for medium-sized devices (landscape tablets and up) with a minimum width of 768px. It modifies the background color of the body selector and adjusts the styles for the .invoice-container, .title-date, .invoice-number, and .space classes. The .invoice-container gets a border, specific width, margin, padding, border radius, and box shadow to create a card-like appearance. The other classes receive adjusted widths.

This will give our invoice generator an upgraded presentation. Create a CSS file with the name of styles.css and paste the given codes into your CSS file. We must create a file with the .css extension.

**styles.css**

**\* {**

**margin: 0;**

**padding: 0;**

**}**

**body {**

**font-family: roboto;**

**background: white;**

**}**

**.material-icons {**

**cursor: pointer;**

**}**

**.invoice-container {**

**margin: auto;**

**padding: 0px 20px;**

**}**

**.invoice-header {**

**display: flex;**

**padding: 70px 0%;**

**width: 100%;**

**}**

**.title {**

**font-size: 18px;**

**letter-spacing: 3px;**

**color: rgb(66, 66, 66);**

**}**

**.date {**

**padding: 5px 0px;**

**font-size: 14px;**

**letter-spacing: 3px;**

**color: rgb(156, 156, 156);**

**}**

**.invoice-number {**

**font-size: 17px;**

**letter-spacing: 2px;**

**color: rgb(156, 156, 156);**

**}**

**.space {**

**width: 50%;**

**}**

**table {**

**table-layout: auto;**

**width: 100%;**

**}**

**table, th, td {**

**border-collapse: collapse;**

**}**

**th {**

**padding: 10px 0px;**

**border-bottom: 1px solid rgb(187, 187, 187);**

**border-bottom-style: dashed;**

**font-weight: 400;**

**font-size: 13px;**

**letter-spacing: 2px;**

**color: gray;**

**text-align: left;**

**}**

**td {**

**padding: 10px 0px;**

**border-bottom: 0.5px solid rgb(226, 226, 226);**

**text-align: left;**

**}**

**.dashed {**

**border-bottom: 1px solid rgb(187, 187, 187);**

**border-bottom-style: dashed;**

**}**

**.total {**

**font-weight: 800;**

**font-size: 20px !important;**

**color: #FF1D89;**

**}**

**input[type=number] {**

**text-align: center ;**

**max-width: 50px;**

**font-size: 15px;**

**padding: 10px;**

**border: none;**

**outline: none;**

**}**

**input[type=text] {**

**max-width: 170px;**

**text-align: left;**

**font-size: 15px;**

**padding: 10px;**

**border: none;**

**outline: none;**

**}**

**input[type=text]:focus {**

**border-radius: 5px;**

**background: #ffffff;**

**box-shadow: 11px 11px 22px #d9d9d9,**

**-11px -11px 22px #ffffff;**

**}**

**input[type=number]:focus {**

**border-radius: 5px;**

**background: #ffffff;**

**box-shadow: 11px 11px 22px #d9d9d9,**

**-11px -11px 22px #ffffff;**

**}**

**input::-webkit-outer-spin-button,**

**input::-webkit-inner-spin-button {**

**-webkit-appearance: none;**

**margin: 0;**

**}**

**/\* Firefox \*/**

**input[type=number] {**

**-moz-appearance: textfield;**

**}**

**.float{**

**width:40px;**

**height:40px;**

**background-color:#FF1D89;**

**color:#FFF;**

**border-radius:100%;**

**text-align:center;**

**box-shadow:**

**0 2.8px 2.2px rgba(0, 0, 0, 0.048),**

**0 6.7px 5.3px rgba(0, 0, 0, 0.069),**

**0 12.5px 10px rgba(0, 0, 0, 0.085),**

**0 22.3px 17.9px rgba(0, 0, 0, 0.101),**

**0 41.8px 33.4px rgba(0, 0, 0, 0.122),**

**0 100px 80px rgba(0, 0, 0, 0.17);**

**}**

**.float:hover {**

**background-color:#ff057e;**

**}**

**.plus{**

**margin-top:10px;**

**}**

**#sum {**

**text-align: right;**

**width: 100%;**

**}**

**#sum input[type=text] {**

**width: 100%;**

**font-size: 33px !important;**

**color: #ff057e;**

**text-align: right !important;**

**}**

**/\* Medium devices (landscape tablets, 768px and up) \*/**

**@media only screen and (min-width: 768px) {**

**body {**

**background: plum;**

**}**

**.invoice-container {**

**border: solid 1px gray;**

**width: 60%;**

**margin: 50px auto;**

**padding: 40px;**

**padding-bottom: 100px;**

**border-radius: 5px;**

**background: white;**

**box-shadow:**

**0 2.8px 2.2px rgba(0, 0, 0, 0.02),**

**0 6.7px 5.3px rgba(0, 0, 0, 0.028),**

**0 12.5px 10px rgba(0, 0, 0, 0.035),**

**0 22.3px 17.9px rgba(0, 0, 0, 0.042),**

**0 41.8px 33.4px rgba(0, 0, 0, 0.05),**

**0 100px 80px rgba(0, 0, 0, 0.07);**

**}**

**.title-date {**

**width: 20%;**

**}**

**.invoice-number {**

**width: 20%;**

**}**

**.space {**

**width: 80%;**

**}**

**}**

**h1 {**

**letter-spacing: 2px;**

**color: plum;**

**}**

**h5 {**

**letter-spacing: 2px;**

**color: gray;**

**}**

**h4 {**

**font-size: 18px;**

**letter-spacing: 2px;**

**color: gray;**

**}**

**p {**

**font-size: 13px;**

**letter-spacing: 2px;**

**color: gray;**

**}**

**.container1 {**

**display: flex;**

**justify-content: space-between; /\* Aligns items to the left and right \*/**

**align-items: center; /\* Centers items vertically \*/**

**font-size: 13px;**

**letter-spacing: 2px;**

**color: gray;**

**padding: 10px 0px;**

**border-bottom: 0.5px solid rgb(226, 226, 226);**

**}**

**Step 3 (JavaScript Code):**

Finally, we need to create a function in JavaScript.

This JavaScript code is responsible for manipulating a table in an HTML document. Let's go through the code step by step to understand its functionality.

The code begins by selecting an element with the ID "table-body" and storing it in the constant variable tBody. This element is expected to be the <tbody> element of an HTML table.

The addNewRow function is defined. It creates a new table row (<tr>) element and assigns the CSS class "single-row" to it. The row's inner HTML is set using a template literal to create the structure of the row, including several <td> elements representing table cells. The last <td> contains a delete button represented by the Material Icons "delete\_outline" icon.

Inside the addNewRow function, the newly created row is inserted before the second-to-last child of the tBody element. This effectively adds the new row just above the last row of the table.

An event listener is added to the element with the ID "add-row". When this element is clicked, it prevents the default form submission behavior and calls the addNewRow function, adding a new row to the table.

The getInput function is defined. It retrieves all the table rows with the CSS class "single-row" using the querySelectorAll method. Then, it retrieves the values entered in the "unit" and "price" input fields for each row. It multiplies these values together to calculate the "amount" and updates the corresponding input field with the calculated value. Finally, it calls the overallSum function.

The overallSum function calculates the total sum of all the "amount" values in the table. It retrieves all the elements with the name attribute "amount" using the getElementsByName method. Then, it iterates over the elements, checks if they have a value, and adds that value to the total variable. Finally, it updates the value of an element with the ID "total" with the calculated total.

Another event listener is added, this time to the tBody element. It listens for click events that occur within the table body. When a click event happens, it checks if the clicked element has an attribute called "action" with a value of "delete". If it does, it calls the delRow function, passing the clicked element as an argument, and then calls the overallSum function.

The delRow function is responsible for removing a row from the table. It receives an element (el) as an argument, representing the clicked delete button. It traverses up the DOM hierarchy to the parent <tr> element and removes it from the DOM.

Create a JavaScript file with the name of script.js and make sure it's linked properly to HTML document, so that the scripts are executed on the page. We create a file with .js extension.

**Script.js**

**const tBody = document.getElementById("table-body");**

**addNewRow =()=> {**

**const row = document.createElement("tr");**

**row.className = "single-row";**

**row.innerHTML = `<td><input type="text" placeholder="Product name" class="product" id="product"></td>**

**<td><input type="number" placeholder="0" name="unit" class="unit" id="unit" onkeyup="getInput()"></td>**

**<td><input type="number" placeholder="0" name="price" class="price" id="price" onkeyup="getInput()"></td>**

**<td><input type="number" placeholder="0" name="amount" class="amount" id="amount" disabled></td>**

**<td style="text-align: right;"><span class="material-icons" action="delete">delete\_outline</span></td>`**

**tBody.insertBefore(row, tBody.lastElementChild.previousSibling);**

**}**

**document.getElementById("add-row").addEventListener("click", (e)=> {**

**e.preventDefault();**

**addNewRow();**

**});**

**//GET INPUTS, MULTIPLY AND GET THE ITEM PRICE**

**getInput =()=> {**

**var rows = document.querySelectorAll("tr.single-row");**

**rows.forEach((currentRow) => {**

**var unit = currentRow.querySelector("#unit").value;**

**var price = currentRow.querySelector("#price").value;**

**amount = unit \* price;**

**currentRow.querySelector("#amount").value = amount;**

**overallSum();**

**})**

**};**

**//Get the overall sum/Total**

**overallSum =()=> {**

**var arr = document.getElementsByName("amount");**

**var total = 0;**

**for(var i = 0; i < arr.length; i++) {**

**if(arr[i].value) {**

**total += +arr[i].value;**

**}**

**document.getElementById("total").value = total;**

**}**

**}**

**//Delete row from the table**

**tBody.addEventListener("click", (e)=>{**

**let el = e.target;**

**const deleteROW = e.target.getAttribute("action");**

**if(deleteROW == "delete") {**

**delRow(el);**

**overallSum();**

**}**

**})**

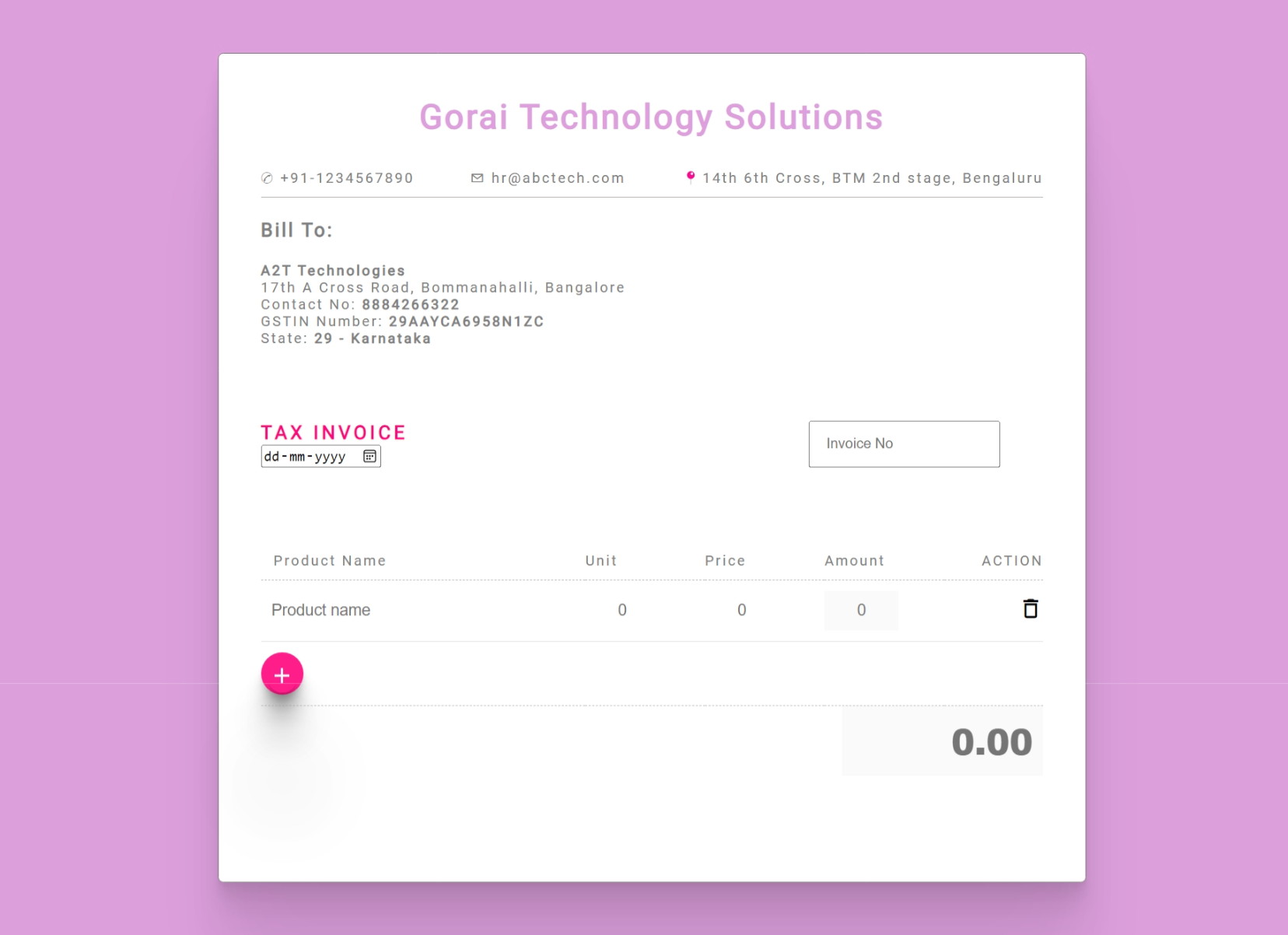
**//Target row and remove from DOM;**

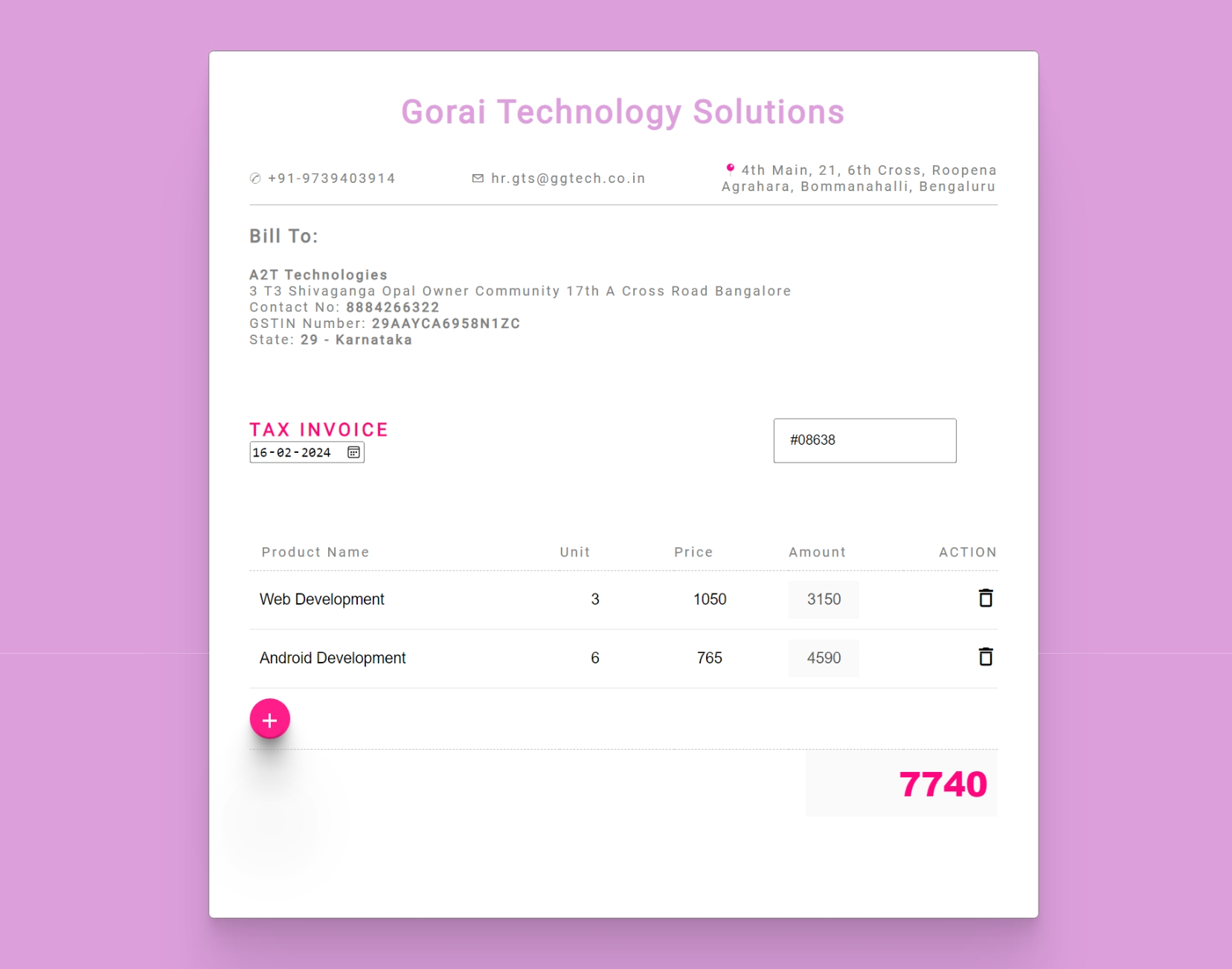
**delRow =(el)=> {**

**el.parentNode.parentNode.parentNode.removeChild(el.parentNode.parentNode);**

**}**

**Final Output:**

****

****