本Web 项目主要围绕药品管理展开，包含用户注册、登录、登出以及药品的添加、编辑、删除和查询等功能。

以下是基于提供的代码完成的前端设计说明：

**1. 整体概述**

本项目前端主要负责与用户进行交互，通过表单、按钮等元素收集用户输入，并向后端发送请求，同时展示后端返回的 JSON 格式数据。设计风格上，整体页面使用Arial字体，背景颜色为#f4f4f4，各功能模块采用白色背景的容器进行布局，搭配适当的阴影和圆角，提升视觉效果。

**2. 页面功能及交互**

该 Web 项目围绕药品管理展开，涉及用户注册、登录、主页药品管理等多个功能模块，各模块的界面功能及交互设计具体如下：

**（1）注册页面**

界面功能：此页面提供给新用户进行账号注册，包含邮箱和密码两个必填输入框以及一个注册按钮，帮助用户创建新的账号以访问系统。

交互流程：用户进入注册页面后，在 “Email” 输入框中输入自己的邮箱地址，在 “Password” 输入框中输入密码。

输入完成后，点击 “Register” 按钮，前端会将用户输入的邮箱和密码整理成 JSON 格式的数据发送至后端服务器，例如：

json

{"email": "user@example.com",

"password": "userpassword"}

后端服务器接收到数据后进行注册处理，如果注册成功，可能会将用户重定向到登录页面；如果注册失败，后端会返回错误信息。

前端接收到错误信息后，会在页面上的.alert类元素中以红色背景的提示框展示具体错误内容，告知用户注册失败的原因。

**（2）登录页面**

界面功能：用户通过该页面输入已注册的邮箱和密码，进行身份验证，验证成功后可进入系统主页。

交互流程：用户访问登录页面，在 “Email” 输入框中输入注册时使用的邮箱，在 “Password” 输入框中输入对应的密码。

点击 “登录” 按钮，前端将邮箱和密码以 JSON 格式数据发送给后端服务器，数据格式与注册时类似：

json

{ "email": "user@example.com",

"password": "userpassword"}

后端服务器对用户输入的信息进行验证，若验证通过，将用户重定向到主页；若验证失败，返回错误信息。

前端接收到错误信息后，会在页面的.alert类元素中以红色背景提示框的形式展示错误内容，提示用户登录失败的原因。

在页面底部有 “没有账号？点此注册” 的注册链接，用户点击该链接可跳转到注册页面。

**（3）主页**

界面功能：主页是系统的核心页面，具备展示用户信息、提供登出功能、支持药品查询以及实现药品管理（添加、编辑、删除）等多种功能。

交互流程：用户信息展示与登出：用户成功登录进入主页后，页面会显示 “欢迎来到主页！” 的欢迎语，同时展示 “您已成功登录。” 的提示以及用户邮箱（{{ user.email }}）。用户若想退出系统，点击 “登出” 按钮，前端向后端发送登出请求，后端处理后将用户重定向到登录页面。

**药品查询：**

用户在 “输入药品名称进行搜索” 的输入框中输入要查询的药品名称关键词。

点击 “查询” 按钮，前端将查询关键词以 JSON 格式数据发送给后端服务器，例如：

json

{"search": "药品名称关键词"}

后端服务器根据接收到的关键词在数据库中进行查询，并将查询结果以 JSON 格式返回给前端。

前端接收到查询结果后，更新页面上的药品列表表格，展示符合查询条件的药品信息。

**药品管理：**

**一．添加药品**：用户点击 “添加药品” 按钮，页面跳转到添加药品页面。在该页面，用户在 “Drug Name” 输入框中输入药品名称，在 “Company” 输入框中输入药品所属公司名称，在 “Quantity” 输入框中输入药品数量，在 “Code” 输入框中输入药品代码。输入完成后，点击 “Add Drug” 按钮，前端将这些信息整理成 JSON 格式数据发送给后端服务器，例如：

json

{"name": "药品名称",

"company": "药品公司",

"quantity": 100,

"code": "药品代码"}

后端服务器接收到数据后执行添加药品操作，并返回操作结果。前端根据返回结果进行相应提示，如添加成功提示 “药品添加成功”，添加失败则在页面上以适当方式展示错误原因。

**二．编辑药品**：在药品列表中，用户点击某一药品对应的 “编辑” 按钮，页面跳转到编辑药品页面（Edit Drug）。该页面的各个输入框会自动填充该药品的现有信息（{{ drug.drug\_name }}、{{ drug.drug\_company }}、{{ drug.drug\_quantity }}、{{ drug.drug\_code }}）。用户可以对这些信息进行修改，修改完成后点击 “Update Drug” 按钮，前端将修改后的信息连同药品的唯一标识（如drug.id）以 JSON 格式数据发送给后端服务器，例如：

json

{ "id": 1,

"drug\_name": "新药品名称",

"drug\_company": "新药品公司",

"drug\_quantity": 200,

"drug\_code": "新药品代码"}

后端服务器接收到数据后执行药品信息更新操作，并返回操作结果。前端根据返回结果在页面上展示相应提示，告知用户编辑操作是否成功。

**三．删除药品**：在药品列表中，用户点击某一药品对应的 “删除” 按钮，前端会向后端服务器发送包含该药品唯一标识（drug.id）的 JSON 格式数据，例如：

json

{"id": 1}

后端服务器接收到请求后执行删除药品操作，并返回操作结果。前端根据返回结果在页面上进行相应提示，告知用户删除操作是否成功，同时更新药品列表，将已删除的药品从列表中移除。

**3. 样式设计**

字体：全局使用Arial字体。

颜色

背景颜色：整体背景为#f4f4f4，内容容器背景为白色。

按钮颜色：默认背景色为#5cb85c，鼠标悬停时变为#4cae4c，文字颜色为白色。

错误提示颜色：提示框背景为#f8d7da，文字颜色为红色。

布局

各页面主体内容使用相应的容器进行包裹，设置了适当的内边距、圆角和阴影效果。

表单元素垂直排列，输入框宽度占满容器，按钮具有合适的尺寸和样式，便于点击操作。

**4. 数据交互**

该 Web 项目的数据交互主要发生在前端与后端之间，涉及用户注册、登录、药品管理等多个功能模块。在交互过程中，前端以 JSON 格式向后端发送请求数据，后端处理后同样以 JSON 格式返回相应结果，以下详细介绍各功能模块的数据交互情况：

**（1）注册功能的数据交互**

**前端请求**：当用户在注册页面输入邮箱和密码并点击 “Register” 按钮后，前端会将用户输入的信息整理成 JSON 格式数据发送给后端：

json

{

"email": "user@example.com",

"password": "userpassword"

}

**后端处理与响应**：后端接收到该 JSON 数据后，会对数据进行验证和处理。验证邮箱是否已被注册、密码是否符合要求等。如果注册过程顺利，后端会将用户信息存储到数据库中，并返回一个表示注册成功的 JSON 响应：

json

{

"status": "success",

"message": "注册成功"

}

若注册过程出现问题，比如邮箱已存在或密码不符合规则，后端会返回包含错误信息的 JSON 响应：

json

{

"status": "failed",

"message": "该邮箱已被注册，请更换邮箱重试"

}

**前端反馈**：前端接收到后端返回的 JSON 响应后，根据status字段判断注册结果。若为 “success”，可能将用户重定向至登录页面；若为 “failed”，则会在页面上以红色背景的.alert提示框展示message字段中的错误信息，告知用户注册失败的具体原因。

**（2）登录功能的数据交互**

**前端请求**：在登录页面，用户输入邮箱和密码并点击 “登录” 按钮，此时前端会将用户输入的登录信息封装成 JSON 格式数据发送给后端，数据结构与注册时类似：

json

{

"email": "user@example.com",

"password": "userpassword"

}

**后端处理与响应**：后端接收到登录请求的 JSON 数据后，会在数据库中查询该邮箱对应的用户记录，并验证密码是否匹配。如果验证通过，后端会生成用户会话信息并返回表示登录成功的 JSON 响应：

json

{

"status": "success",

"message": "登录成功"

}

若邮箱不存在或密码错误，后端会返回包含错误信息的 JSON 响应：

json

{

"status": "failed",

"message": "邮箱或密码错误，请重新输入"

}

**前端反馈**：前端根据后端返回的 JSON 响应中的status字段判断登录结果。若登录成功，会将用户重定向至主页；若登录失败，会在页面的.alert提示框中展示message字段的错误信息，提醒用户登录失败并说明原因。

**（3）主页相关功能的数据交互**

**药品查询**

**前端请求**：用户在主页的药品查询输入框中输入要查询的药品名称关键词，点击 “查询” 按钮后，前端将查询关键词整理成 JSON 格式数据发送给后端：

json

{

"search": "药品名称关键词"

}

**后端处理与响应：**后端接收到查询请求的 JSON 数据后，会根据关键词在药品数据库表中进行查询操作。查询完成后，将查询结果以 JSON 格式返回给前端。如果查询到符合条件的药品，响应数据如下：

json

{

"status": "success",

"message": "查询成功",

"data": [

{

"drug\_name": "药品1名称",

"drug\_company": "公司1名称",

"drug\_quantity": 100,

"drug\_code": "药品1代码"

},

{

"drug\_name": "药品2名称",

"drug\_company": "公司2名称",

"drug\_quantity": 50,

"drug\_code": "药品2代码"

}

]

}

若没有查询到符合条件的药品，后端返回的 JSON 响应：

json

{

"status": "success",

"message": "没有找到符合条件的药品",

"data": []

}

**前端反馈**：前端接收到后端返回的 JSON 响应后，根据status字段判断查询结果。若查询成功，会根据data字段中的内容更新页面上的药品列表表格，展示查询到的药品信息；若data为空，则在页面上显示提示信息告知用户没有找到符合条件的药品。

**药品添加**

**前端请求**：在添加药品页面，用户输入药品的名称、公司、数量、代码等信息后，点击 “Add Drug” 按钮，前端会将这些信息封装成 JSON 格式数据发送给后端：

json

{

"name": "药品名称",

"company": "药品公司",

"quantity": 100,

"code": "药品代码"

}

**后端处理与响应**：后端接收到添加药品的 JSON 数据后，会对数据进行验证并将药品信息插入到数据库中。如果添加成功，后端返回的 JSON 响应：

json

{

"status": "success",

"message": "药品添加成功"

}

若添加过程中出现问题，比如药品代码重复等，后端会返回包含错误信息的 JSON 响应：

json

{

"status": "failed",

"message": "药品代码已存在，请更换代码后重试"

}

**前端反馈：**前端根据后端返回的 JSON 响应中的status字段判断添加结果。若添加成功，可能会在页面上显示提示信息告知用户添加成功，或者将用户重定向回主页；若添加失败，则在页面上以适当方式展示message字段中的错误信息，帮助用户解决问题。

**药品编辑**

**前端请求**：在编辑药品页面，用户对药品的名称、公司、数量、代码等信息进行修改后，点击 “Update Drug” 按钮，前端会将修改后的信息连同药品的唯一标识（如drug.id）整理成 JSON 格式数据发送给后端：

json

{

"id": 1,

"drug\_name": "新药品名称",

"drug\_company": "新药品公司",

"drug\_quantity": 200,

"drug\_code": "新药品代码"

}

**后端处理与响应**：后端接收到编辑药品的 JSON 数据后，会根据药品的唯一标识在数据库中找到对应的药品记录，并更新相关信息。如果编辑成功，后端返回的 JSON 响应：

json

{

"status": "success",

"message": "药品信息更新成功"

}

若编辑过程出现问题，后端会返回包含错误信息的 JSON 响应：

json

{

"status": "failed",

"message": "更新失败，可能是数据格式有误"

}

**前端反馈**：前端根据后端返回的 JSON 响应中的status字段判断编辑结果。若编辑成功，可能在页面上显示成功提示信息，或者将用户重定向回主页并更新药品列表；若编辑失败，会在页面上展示message字段的错误信息，协助用户排查问题。

**药品删除**

**前端请求**：在主页的药品列表中，用户点击某药品对应的 “删除” 按钮，前端会将该药品的唯一标识（drug.id）封装成 JSON 格式数据发送给后端：

json

{

"id": 1

}

**后端处理与响应**：后端接收到删除药品的 JSON 数据后，会根据药品的唯一标识在数据库中找到对应的记录并删除。如果删除成功，后端返回的 JSON 响应：

json

{

"status": "success",

"message": "药品删除成功"

}

若删除过程出现问题，后端会返回包含错误信息的 JSON 响应：

json

{

"status": "failed",

"message": "删除失败，该药品可能正在被使用"

}

**前端反馈：**前端根据后端返回的 JSON 响应中的status字段判断删除结果。若删除成功，会在页面上更新药品列表，移除已删除的药品记录，并可显示提示信息告知用户删除成功；若删除失败，则在页面上展示message字段的错误信息，告知用户删除失败的原因。

**5. 响应式设计**

响应式设计能够让网页在不同设备和屏幕尺寸下都能提供良好的用户体验，确保页面布局合理、内容可读且操作方便。该 Web 项目在响应式设计方面采取了多种策略，以下从视口设置、容器布局、输入框和按钮设计等方面详细阐述：

**（1）视口设置**

在每个 HTML 页面的<head>标签中，都设置了以下meta标签：

html

<meta name="viewport" content="width=device - width, initial - scale=1.0">

这一设置是响应式设计的基础，它的作用是让页面的宽度自动适应设备的屏幕宽度，并且初始缩放比例为 1.0。这样，无论用户使用的是桌面电脑、平板电脑还是手机，页面都能根据设备屏幕尺寸进行适配，避免出现页面过宽或过窄、字体过小等问题，从而为用户提供较好的初始浏览体验。

**（2）容器布局**

宽度设置：项目中的多个容器都进行了灵活的宽度设置。例如，在登录页面和注册页面中，.login - container和.register - container的宽度被设置为固定的 300px。这在小屏幕设备（如手机）上，能保证内容集中展示，不会因屏幕过窄而导致布局混乱；在较大屏幕设备上，也不会占据过多空间。

而主页中的.container容器则采用了相对宽度和最大宽度限制相结合的方式，宽度设置为 85%，同时设置max - width: 1000px。这使得容器在小屏幕设备上能充分利用屏幕空间，随着屏幕宽度的增加，容器宽度会按比例增大，但最大不会超过 1000px，从而在大屏幕设备上保证内容不会过于分散，维持良好的视觉效果和布局合理性。

在编辑药品页面的.edit - container容器，宽度设置为视口宽度的 90%（width: 90%），同时限制最大宽度为 600px（max - width: 600px）。这种设置能在小屏幕设备上提供较大的可操作区域，在大屏幕设备上又能控制容器大小，保持页面的整体协调性。

高度设置：部分页面的<body>元素设置了min - height: 100vh，确保页面内容至少和视口一样高。这在一些内容较少的页面中，能保证页面整体布局在不同屏幕高度下的完整性，不会出现内容集中在页面顶部，下方留有大片空白的情况。例如在登录和注册页面，即使内容不多，整个页面也能均匀地展示在屏幕中间。

**（3）输入框和按钮设计**

输入框：各类输入框（如文本输入框、数字输入框等）都设置了width: 100%，使其能填满所在容器的宽度。这种设计在不同屏幕尺寸下都能保证输入框有足够的宽度，方便用户输入信息。例如在手机屏幕上，输入框能充分利用有限的屏幕宽度，在平板电脑和桌面电脑上也能合理占据空间，不会显得过小或过大。

按钮：按钮在不同屏幕尺寸下也保持了较好的可操作性。按钮设置了合适的内边距（如padding: 10px），使其在小屏幕设备上也能方便用户点击。同时，按钮的样式在鼠标悬停时会有变化（如背景颜色改变），这种交互效果在不同设备上都能正常展示，增强了用户与页面的交互体验。

**（4）整体布局**

弹性布局（Flexbox）：大量使用了 Flexbox 布局来实现页面元素的排列和对齐。例如在主页中，<body>元素设置了display: flex; justify - content: center; align - items: center; flex - direction: column;，使得页面内容垂直排列并在水平和垂直方向上居中对齐。这种布局方式在不同屏幕尺寸下都能灵活调整元素的位置和间距，保证页面布局的合理性。在较小屏幕上，内容会紧凑排列，在大屏幕上则能均匀分布，提高了页面的可读性和美观性。

表格展示：在主页展示药品列表的表格部分，设置了width: 100%和border - collapse: collapse，确保表格能适应容器宽度，并在不同屏幕尺寸下保持紧凑的布局。同时，表头和表行设置了适当的内边距（padding: 10px），保证数据在不同屏幕上都能清晰展示。对于表行的鼠标悬停效果（tr:hover），也能在不同设备上正常呈现，提升了用户查看数据的交互体验。

通过上述响应式设计策略的综合应用，该 Web 项目能够在多种设备和屏幕尺寸下保持良好的用户体验，确保页面的可用性和美观性。