گروه ۵

نام پروژه: Simple To-Do

Extension in VS Code: simple-to-do

Github : https://github.com/TMhamid/Simple-To-Do.git

اعضا گروه:

۱	**محمدیاسین سنجری پور**
۲	**حمیدرضا کناری زاده**
۳	**مهرپویا خسروی**
۴	**الینا حیدری**

ساختار این پیوست شامل مراحل زیر است:

- **نیازمندی ها و تحلیل**
- **طراحی**
- **پیاده سازی**
- **استقرار**

- **برنامه ریزی**
- **چالش ها**

نیازمندی ها و تحلیل

برای شناسایی و مستندسازی نیازمندی‌ها، می‌توانیم نیازمندی‌های کارکردی (Functional Requirements)و غیرکارکردی (Non-functional Requirements) پروژه را بررسی و در قالب‌های مناسب مستند کنیم.

1.نیازمندی‌های کارکردی(Functional Requirements)

این نیازمندی‌ها به نحوه عملکرد افزونه و ویژگی‌های آن مربوط می‌شوند. در این بخش، وظایف و فعالیت‌های مشخصی که افزونه باید انجام دهد، ذکر می‌شود.

نیازمندی‌های کارکردی افزونه "مدیریت وظایف" به شرح زیر است:

- **مدیریت وظایف**:

۱. **افزودن وظیفه**:

- کاربر باید قادر باشد وظیفه جدیدی وارد کند.
- وقتی کاربر وظیفه‌ای اضافه می‌کند، این وظیفه باید به لیست وظایف افزوده شود و به Webview ارسال شود.

۲. **تغییر وضعیت انجام‌شده**:

- کاربر باید قادر باشد روی دایره کنار هر وظیفه کلیک کند تا وضعیت "انجام‌شده" یا "انجام‌نشده" آن تغییر کند.
- پس از تغییر وضعیت، وضعیت جدید باید به Webview ارسال شود.

۳. **حذف وظایف**:

- کاربر باید بتواند وظایف را حذف کند.
- کاربر باید بتواند یک وظیفه خاص را انتخاب کرده و حذف کند.

- کاربر باید بتواند تمامی وظایف را به طور همزمان حذف کند.

٤. **بارگذاری مجدد وظایف:**

- کاربر باید قادر باشد با استفاده از دکمه"Reload Tasks" ، وظایف را مجدداً بارگذاری کند.
- پس از بارگذاری مجدد، وظایف جدید و به روز به Webview ارسال می‌شود.

داستان‌های کاربر:

١. **افزودن وظیفه:**

- **به عنوان یک کاربر،** من می‌خواهم بتوانم یک وظیفه جدید وارد کنم تا آن را در لیست وظایف مشاهده کنم.
- **معیار پذیرش** :وظیفه جدید باید پس از وارد کردن، به صورت موفقیت‌آمیز در Webview نمایش داده شود.

٢. **تغییر وضعیت وظیفه:**

- **به عنوان یک کاربر،** من می‌خواهم وضعیت انجام‌شده یک وظیفه را تغییر دهم تا بتوانم وظیفه‌هایی که انجام داده‌ام را از بقیه متمایز کنم.
- **معیار پذیرش** :وضعیت وظیفه پس از کلیک روی دایره مربوطه باید تغییر کند و در Webview به روز شود.

٣. **حذف وظیفه:**

- **به عنوان یک کاربر،** من می‌خواهم یک وظیفه خاص را از لیست حذف کنم تا وظایف انجام‌نشده یا غیرضروری از بین بروند.
- **معیار پذیرش** :وظیفه حذف‌شده باید از لیست Webview ناپدید شود.

٤. **حذف همه وظایف:**

- **به عنوان یک کاربر،** من می‌خواهم تمامی وظایف را از لیست حذف کنم تا زمانی که هیچ وظیفه‌ای برای نمایش وجود نداشته باشد.

- **معیار پذیرش** :پس از حذف همه وظایف، پیام "No tasks yet" باید در Webviewنمایش داده شود.

.2نیازمندی‌های غیرکارکردی(Non-functional Requirements)

این نیازمندی‌ها به ویژگی‌هایی مانند عملکرد، امنیت، قابلیت دسترس‌پذیری، و استفاده‌پذیری افزونه مربوط می‌شوند. این نیازمندی‌ها به نحوه عملکرد کلی افزونه و نحوه استفاده از آن توسط کاربران اشاره دارند.

نیازمندی‌های غیرکارکردی:

۱. **عملکرد:**

- افزونه باید بتواند وظایف را به سرعت اضافه، حذف و به روز کند.
- افزونه باید قادر باشد پس از هر تغییر در لیست وظایف (افزودن، حذف یا تغییر وضعیت)، تغییرات را بلافاصله در Webview نمایش دهد.

۲. **قابلیت دسترس‌پذیری:**

- افزونه باید قابلیت دسترس‌پذیری بالا برای تمامی کاربران را فراهم کند.
- ورودی‌های متنی باید دارای aria-labelباشند تا برای کاربران با مشکلات بینایی قابل استفاده باشد.
- دکمه‌ها باید قابل دسترسی با صفحه‌کلید و همچنین از طریق فناوری‌های کمکی مانند صفحه‌خوان‌ها قابل تشخیص باشند.

۳. **امنیت:**

- افزونه باید از امنیت داده‌ها و اطلاعات کاربران محافظت کند.
- ارتباطات میان Webview و افزونه باید به گونه‌ای باشد که از حملات مخرب) مانند (XSS محافظت شود.

٤. **پایداری:**

- افزونه باید بتواند به طور مداوم بدون کرش کردن یا افت عملکرد در هر بار استفاده، وظایف را مدیریت کند.

٥. **کارایی:**

- افزونه باید برای حجم بالای وظایف به طور بهینه عمل کند و عملکرد آن تحت بار زیاد تحت تاثیر قرار نگیرد.

۶. **طراحی رابط کاربری**:

- طراحی رابط کاربری باید ساده، کاربرپسند و واضح باشد.
- استفاده از رنگ‌ها و اندازه‌ها باید به گونه‌ای باشد که همگان بتوانند وظایف خود را به راحتی مشاهده و مدیریت کنند.

انتظارات از بخش تحلیل شامل:

برای تحلیل پروژه و درک بهتر ساختار سامانه، ابتدا باید اجزای مختلف سامانه را شناسایی کرده و نحوه تعامل این اجزا را توضیح دهیم. سپس، فرآیندها و تعاملات سامانه را بررسی کرده و مستنداتی برای درک بهتر نیازمندی‌ها و طراحی سیستم ارائه خواهیم کرد.

شناسایی اجزای مختلف سامانه

افزونه‌ای که در این پروژه در حال توسعه است، شامل چندین جزء اصلی است که در تعامل با یکدیگر قرار دارند. اجزای مختلف سامانه به شرح زیر هستند:

۱. **Frontend**

- ویژگی‌ها :این جزء وظیفه نمایش وظایف به کاربر را دارد Webview .از HTML، CSSو JavaScript برای نمایش لیست وظایف و تعامل با کاربر استفاده می‌کند.
- فرآیندها:
  - دریافت وظایف از افزونه (پس از هر تغییر در وظایف).
  - ارسال درخواست‌های کاربر برای تغییر وضعیت وظایف (اضافه کردن، حذف کردن یا تغییر وضعیت).

۲. Backend

- ویژگی‌ها :این قسمت مسئول پردازش منطق اصلی افزونه است و با Webview در ارتباط است.

- مدیریت وظایف :این جزء وظیفه مدیریت داده‌ها (وظایف) را بر عهده دارد، از جمله افزودن، حذف و تغییر وضعیت وظایف.
- ذخیره‌سازی داده‌ها :وظایف به طور موقت ذخیره می‌شوند و پس از هر تغییر وضعیت یا وظیفه جدید، داده‌ها به Webview ارسال می‌شود.

- فرآیندها:
  - ارسال و دریافت داده‌ها بین Webview و افزونه.
  - ذخیره وضعیت وظایف و تغییرات آن‌ها در حافظه موقت افزونه.

۳. API

- ویژگی‌ها API :رابطی است که بین Webview و Backend برقرار می‌کند. این API از طریق درخواست‌های HTTP اطلاعات را از وب‌ویو به Backend ارسال می‌کند و بالعکس.

- فرآیندها:
  - ارسال درخواست برای افزودن، حذف یا تغییر وضعیت وظایف.
  - دریافت پاسخ‌ها و ارسال داده‌ها به.Webview

٤. Data Structure

ویژگی‌ها :اطلاعات مربوط به وظایف در یک ساختار داده‌ای (مانند آرایه یا شیء) ذخیره می‌شود. هر وظیفه باید اطلاعاتی مانند عنوان وظیفه، وضعیت (انجام‌شده یا انجام‌نشده) و شناسه (ID) داشته باشد.

- فرآیندها:
  - وظایف به صورت یک آرایه از اشیاء ذخیره می‌شوند و به‌روزرسانی‌های آن‌ها در این آرایه اعمال می‌شود.
  - داده‌ها به Webview ارسال می‌شود تا وظایف به روز شده را نمایش دهد.

2.نحوه تعامل اجزا

۱. **Webview**و**Backend**

- Webviewاز طریق API با Backend ارتباط برقرار می‌کند.

- عملیات اضافه کردن وظیفه :زمانی که کاربر وظیفه جدیدی را وارد می‌کند، این درخواست از Webview به Backend ارسال می‌شود و پس از پردازش در Backend، وظیفه به WebZZZiew باز می‌گردد و در لیست وظایف نمایش داده می‌شود.
- عملیات تغییر وضعیت وظیفه :وقتی کاربر وضعیت یک وظیفه را تغییر می‌دهد (مثلاً از "انجام‌نشده" به "انجام‌شده")، این درخواست از Webview به Backendارسال می‌شود و وضعیت جدید در Backend به روز می‌شود. سپس این تغییرات به Webview ارسال می‌شود تا در لیست وظایف نمایش داده شود.
- عملیات حذف وظیفه :زمانی که کاربر یک وظیفه را حذف می‌کند، این درخواست از Webview به Backend ارسال می‌شود و وظیفه حذف‌شده از حافظه افزونه و لیست Webview پاک می‌شود.

۲. Backendو ساختار داده‌ها:

- درBackend ، داده‌های وظایف در یک ساختار داده‌ای (آرایه یا شیء) ذخیره می‌شود.
- هر بار که تغییرات (افزودن، حذف یا تغییر وضعیت) بر روی وظایف انجام می‌شود، این تغییرات در ساختار داده‌ها اعمال می‌شود.
- ساختار داده‌ها می‌تواند شامل فیلدهایی مانند"ID" ، "عنوان وظیفه"، "وضعیت" و "تاریخ ایجاد" باشد.

۳. APIوWebview

- APIنقش رابط بین Webview و Backend را ایفا می‌کند.
- درخواست‌های کاربر از Webview به API ارسال می‌شود، که این درخواست‌ها شامل عملیات‌هایی مانند اضافه کردن، حذف کردن یا تغییر وضعیت وظایف هستند.
- APIدرخواست‌ها را به Backend ارسال می‌کند و پس از پردازش، نتایج را به Webviewباز می‌گرداند.

.3فرآیندهای سامانه

اضافه کردن وظیفه:

- کاربر با استفاده ازWebview ، وظیفه‌ای جدید را وارد می‌کند.

- Webviewدرخواست افزودن وظیفه را به API ارسال می‌کند.
- APIدرخواست را به Backend ارسال می‌کند.
- Backendوظیفه را به ساختار داده‌ها اضافه می‌کند و وضعیت جدید وظیفه را به Webviewارسال می‌کند.

تغییر وضعیت وظیفه:

- کاربر وضعیت یک وظیفه را تغییر می‌دهد (مثلاً از "انجام‌نشده" به "انجام‌شده").
- Webviewدرخواست تغییر وضعیت را به API ارسال می‌کند.
- APIدرخواست را به Backend ارسال می‌کند.
- Backendوضعیت وظیفه را به روز می‌کند و وضعیت جدید به Webview ارسال می‌شود.

حذف وظیفه:

- کاربر وظیفه‌ای را انتخاب می‌کند که آن را حذف کند.
- Webviewدرخواست حذف وظیفه را به API ارسال می‌کند.
- APIدرخواست حذف را به Backend ارسال می‌کند.
- Backendوظیفه را از ساختار داده‌ها حذف می‌کند و وضعیت جدید را به Webviewارسال می‌کند.

توضیح افزونه

هدف از توسعه

هدف اصلی از توسعه این افزونه، بهبود بهره‌وری و فرآیند مدیریت و سازماندهی کارهای روزانه و پروژه‌های برنامه‌نویسان در محیط **Visual Studio Code**است. افزونه‌ها به‌طور خاص برای توسعه‌دهندگان طراحی می‌شوند تا کارهای مختلفی را برای مدیریت و بررسی خود بر روی پروژه‌های برنامه‌نویسی حفظ کنند. در واقع، این افزونه به کاربران

این امکان را می‌دهد که بدون نیاز به جابه‌جایی میان ابزارهای مختلف، همه وظایف خود را در محیط توسعه دنبال کن.

مسائلی که قرار است توسط افزونه حل شود

- **مدیریت و انجام وظایف** :بسیاری از برنامه‌نویسان ممکن است در پروژه‌های خود وظایف متعددی داشته باشند که مدیریت آن‌ها به صورت دستی می‌تواند زمان‌بر و ممکن باشد. این امکان را فراهم می‌کند که به‌راحتی ثبت و مدیریت شود.
- **یکپارچگی با محیط توسعه** :به‌طور معمول، برنامه‌نویسان برای انجام وظایف خود از **Trello**یا **Todoist**استفاده می‌کنند. این افزونه نیاز به جابه‌جایی بین محیط‌های مختلف را از بین می‌برد و تمامی فرآیندهای مدیریت وظایف را در داخل **VS Code**انجام می‌دهد.
- **حفظ تمرکز و بهبود بهره‌وری** :از آنجایی که این افزونه مستقیماً در محیط کدنویسی قرار می‌گیرد، کاربران می‌توانند هم‌زمان با نوشتن کد و مدیریت کارها بپردازند، بدون اینکه باید خود را از دست بدهند یا از محیط توسعه خارج کنند.
- **بهبود اولویت‌بندی و زمان‌بندی** :برنامه‌نویسان ممکن است گاهی از اولویت‌بندی و زمان‌بندی تعیین وظایف خود غافل شوند. این امکان را فراهم می‌کند که کارها بر اساس اولویت و تاریخ انجام، به طور مستقیم مدیریت می‌شوند.

ویژگیهای کلیدی افزونه برای کاربران

- **مدیریت وظایف** :کاربران می‌توانند اضافه کنند، وظایف موجود را ویرایش یا حذف کنند. همچنین می‌توان توضیحات و جزئیات مربوط به هر وظیفه را درج کرد.
- **تغییر وضعیت وظایف** :هر وظیفه می‌تواند وضعیت‌های مختلف را داشته باشد
- **دسته و اولویت‌بندی وظایف** :وظایف می‌توانند بر اساس اولویت‌های مختلف دسته‌بندی شوند. افزونه های امکان تنظیم وظایف بر اساس تاریخ، اولویت و وضعیت را به کاربران می دهد.
- **نمایش لیست وظایف** :افزونه یک لیست وظایف کاربر را به صورت بصری نمایش می دهد. این لیست شامل تمام جزئیات مانند تاریخ ایجاد، تاریخ پایان، وضعیت و اولویت خواهد بود.
- **ذخیره‌سازی محلی** :همه وظایف به‌صورت محلی ذخیره می‌شوند، به این معناست که حتی پس از بستن محیط **VS Code**، وظای

- **یکپارچگی با VS Code** : افزونه به‌طور کامل در محیط توسعه **VS Code** عمل می‌کند. کاربران نیازی به استفاده از ابزارهای خار
- **رابط کاربری ساده و کاربرپسند** :افزونه با یک رابط کاربری ساده و شهودی طراحی شده است که برای تمامی کاربران، از مبتدی تا حرفه ای، قابل استفاده است.
- **پشتیبانی از پروژه** :این امکان را به کاربران می دهد که برای هر پروژه وظایف خاص خود را انجام دهند

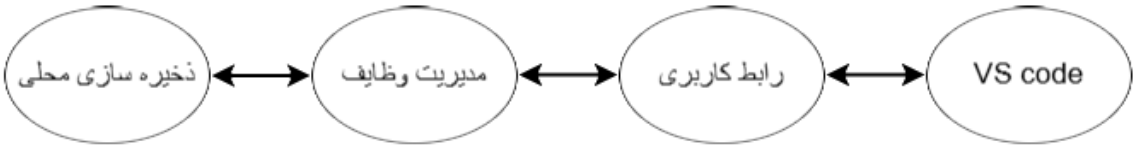
این ویژگی‌ها باعث می‌شوند که افزونه‌های ابزار مفید برای همه کسانی باشند که به دنبال مدیریت و سازماندهی بهتر زمان و وظایف خود در محیط توسعه باشند.

مدل مفهومی

مدل مفهومی برای نشان دادن ارتباطات مفهومی بین اجزای سامانه به‌کار می‌رود. در این مدل، اجزای اصلی افزونه و نحوه طراحی آنها به صورت گرافیکی

- **افزونه مدیریت وظایف** به‌طور مستقیم با بخش‌هایی از **Visual Studio Code** مانند **پنل کنسول** ، **تسک‌بار** ، و **نمایش فایل‌های** ارتباطی دارد.
- اجزای اصلی مفهومی این افزونه عبارتند از:
  - **مدیریت وظایف** :این جزء شامل تمامی عملیات ثبت، ویرایش و حذف وظایف است.
  - **سیستم ذخیره‌سازی محلی** :این سیستم وظایف را ذخیره می‌کند.
  - **ارتباط کاربری** :ارتباطی که کاربران از طریق آن با افزونه دارند.

مدل مفهومی:



نمودار کلی جریان داده(Context Diagram)

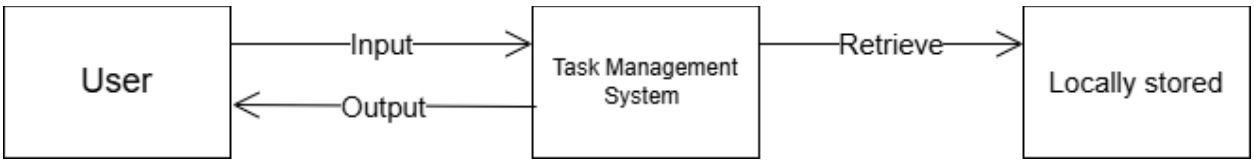
این سطح نشان‌دهنده ارتباطات کلی سیستم با عوامل خارجی است.

اجزای اصلی:

۱. کاربر :(User) داده‌ها را وارد می‌کند و نتایج را مشاهده می‌کند.

۲. سیستم مدیریت وظایف :(Task Management System) داده‌ها را پردازش و ذخیره می‌کند.

۳. پایگاه داده :(Database) وظایف را ذخیره و بازیابی می‌کند.



نقشه معماری لایه‌ای (نمودار معماری لایه‌ای)

این نقشه‌ها، سیستم را به لایه‌های مجزا تقسیم می‌کند و نشان می‌دهد هر لایه چگونه با دیگری در میان است.

لایه ها:

۱. نمایش لایه (لایه ارائه):

- اجزای رابط کاربری مانند فرم ها و نمایش لیست وظایف.

۲. لایه منطق کسب و کار:(Business Logic Layer)

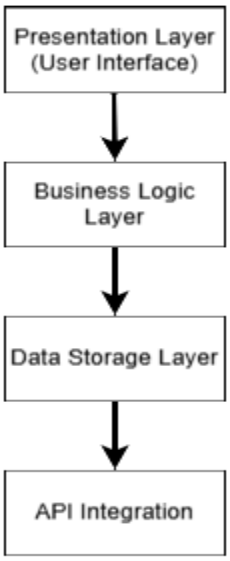
- مسئول پردازش و مدیریت داده ها.

۳. لایه ذخیره‌سازی (لایه ذخیره‌سازی داده):

- نگهداری داده ها در حافظه یا پایگاه داده.

٤. API :

- مسئول ارسال و دریافت اطلاعات به/از سرور.



نمودارهای توالی(Sequence Diagrams)

نمودارهای توالی برای نمایش **ترتیب عملیات** و **تعامل بین اجزای سیستم** در یک سناریوی خاص استفاده می‌شوند. در ادامه یک نمودار توالی برای افزونه مدیریت وظایف ارائه شده است.

سناریو: ایجاد و ذخیره یک وظیفه توسط کاربر

توضیح سناریو:

۱. کاربر اطلاعات یک وظیفه جدید را از طریق رابط کاربری وارد می‌کند.

۲. سیستم اطلاعات واردشده را اعتبارسنجی می‌کند.

۳. اگر داده‌ها معتبر باشند، وظیفه به پایگاه داده ارسال و ذخیره می‌شود.

٤. سیستم پیام تأیید ذخیره موفق را به کاربر نمایش می‌دهد.

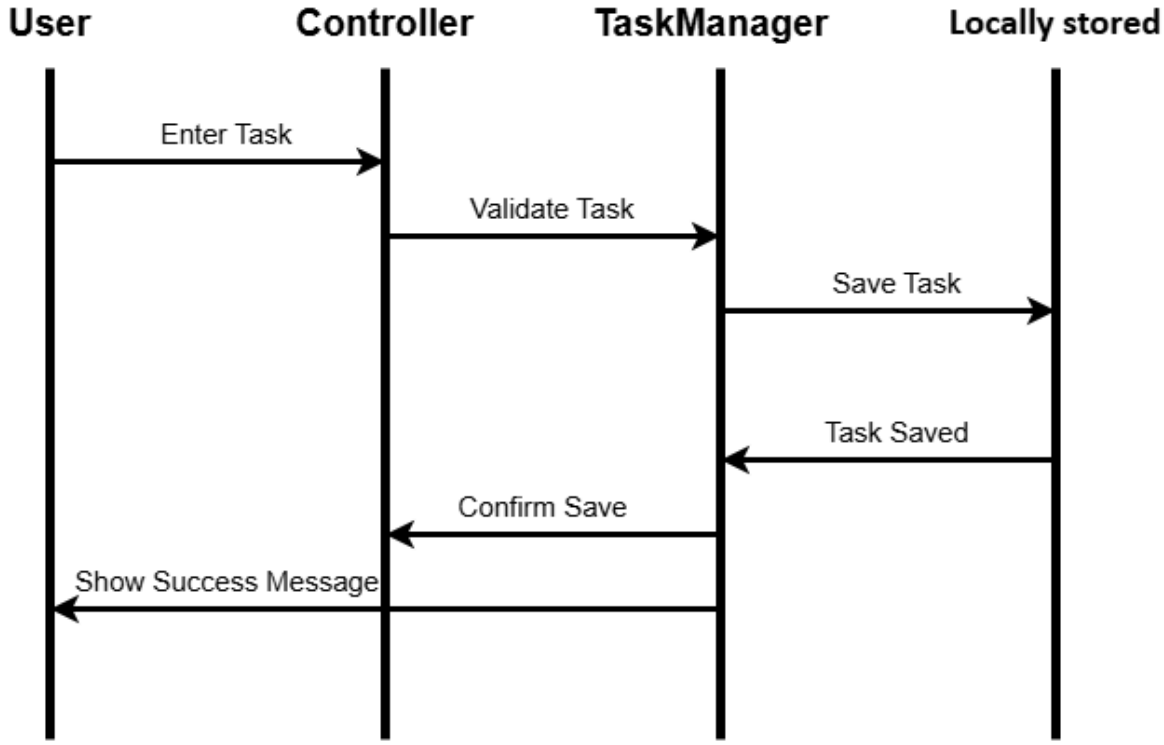
اجزای دخیل در سناریو:

۱. **User**کاربر که اطلاعات را وارد می‌کند.

۲. **Controller**رابط کاربری که داده‌ها را دریافت و به سیستم ارسال می‌کند.

۳. **TaskManager**مسئول مدیریت منطق کسب‌وکار و اعتبارسنجی داده‌ها.

٤. **Locally stored**پایگاه داده برای ذخیره وظایف.



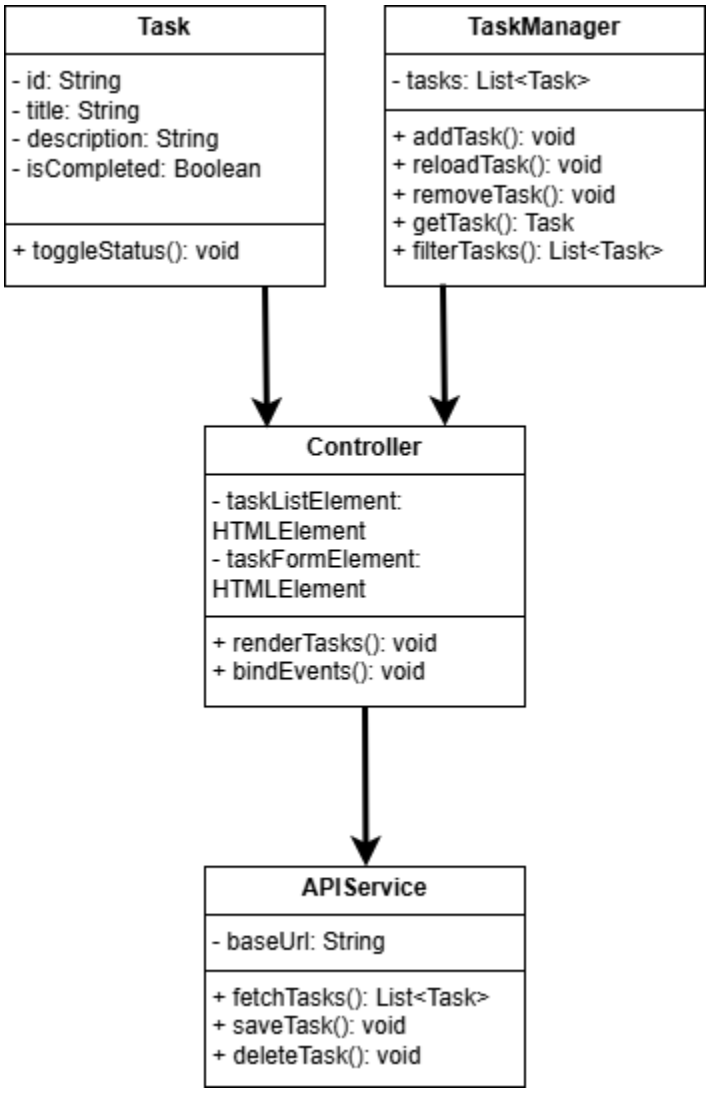
Enter Task کاربر اطلاعات وظیفه جدید را از طریق رابط کاربری وارد می‌کند.

Validate Task رابط کاربری اطلاعات وظیفه را برای اعتبارسنجی به TaskManagerارسال می‌کند.

Save Task پس از اعتبارسنجی موفق، TaskManager داده‌ها را به پایگاه داده ارسال می‌کند.

Task Saved پایگاه داده تأیید ذخیره موفق وظیفه را به TaskManager بازمی‌گرداند.

نمودار کلاس(Class Diagram)



نمودار ترتیبی(Sequence Diagram)

اجزای شرکت‌کننده:

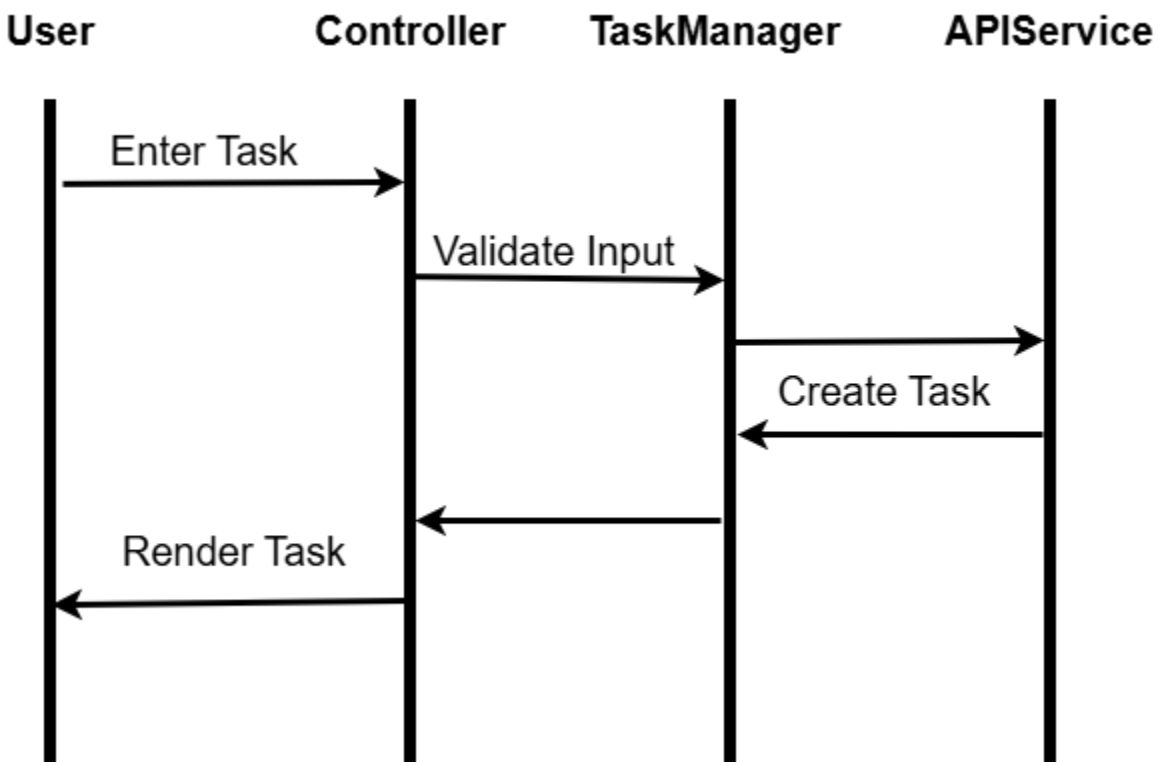
۱. User کاربر :تعامل اولیه را آغاز می‌کند.

۲. Controller رابط کاربری را مدیریت می‌کند.

۳. TaskManager وظایف را مدیریت و ذخیره می‌کند.

۴. APIService وظایف را در پایگاه داده یا سرور ذخیره می‌کند.

نمودار ترتیبی(Sequence Diagram)



استقرار

https://code.visualstudio.com/api/working-with-extensions/publishing-extension#publishing-extensions

برنامه ریزی

- **تقسیم وظایف**

۱. **محمد یاسین سنجری پور:** مسئول Trello ، مسئول فایل اجرایی پروژه ، مسئول گیت و گیت هاب
۲. **حمیدرضا کناری زاده:** مسئول فایل مدیریت task ها ، مسئول فرانت اند افزونه ، نصب و راه اندازی افزونه ، API VS Code
۳. **مهرپویا خسروی:** Scrum master ، مسئول محتوا آموزشی ، مسئول API VS Code ، مسئول تحلیل و طراحی محصول، مدیریت task ها
۴. **الینا حیدری:** مسئول UI و UX ، فایل اجرایی پروژه ، بررسی کد سورس های دیگر افزونه ها ، محتوای آموزشی

مستند سازی جلسات

- **جلسه اول:** در این جلسه بچه با هم درباره موضوع پروژه بحث کردن نظر اکثریت روی موضوع مدیریت برنامه بود پروژه نهایی یک افزونه چک لیست شد.
بحث بر این بود اول بتوانیم ویژگی های مهم انجام بدیم و در صورت نیاز موارد دیگر را اضافه کنیم در این مورد **یاسین سنجری** مسئول موارد شبیه به چک لیست شد.
- **جلسه دوم:** بحث موارد اموزشی بود که چه مواردی نیازه و مسئول برای ان تعریف بشود و هر ورد تمرکز کافی از کاری میخواد انجام بده داشته باشد.
الینا حیدری مسئول طراحی UI وUX بود که به فرانت پروژه مربوط بود. تحلیل و طراحی پروژه و اسکرام مستر برعهده **مهرپویا خسروی** شد همین موارد درون

پروژه شامل بود ولی ویژگی بشود تسک ها در دیتابیس ذخیره بشود در موارد دیگر انجام شود.

- **جلسه سوم:** در این جلسه مسئولیت های قبلی انجام شد و نتایج در جلسه بحث شد و رابط کاربری با فرانت کار بحث شد که مسئول قسمت فرانت اند **حمیدرضا کناری زاده** است. طراحی و نیازمندی ها به اشتراک گذاشته شد.
- **جلسه چهارم:** در این جلسه بچه مواد اموزشی را مورد بحث قرار دادن روند مثبتی بود.
- **جلسه پنجم:** در این قسمت فایل اجرایی پروژه انجام شد که مسئول آن **یاسین سنجری** بود در کنارش **الینا حیدری** همکاری می کرد و بقیه اعضا گروه مواد لازم کد را بررسی میکردن در زمینه API و مدیریت تسک ها آیا خروجی مدنظرهست یا نه.
- **جلسه ششم:** بحث فایل مدیریت تسک ها مسئول **حمیدرضا کناری زاده** و در کنارش **مهرپویا خسروی** بود تا تحلیل نیازمندی های پروژه به درستی انجام شود در در همکاری رفع باگ یکی از کار هایی باید انجام دهد تا تایم هدر نرود.
- **جلسه هفتم:** **الینا حیدری** مسئول طراحی UX, UI و مسئول **حمیدرضا کناری زاده** فرانت اند می باشد و فایل پروژه مربوط به این را انجام دادند و اعضا دیگر این فایل را بررسی کردن. که مشکل در اجرا به وجود امد که در بحث شد. که باگ آن رفع شد .
- **جلسه هشتم:** کار های مربوط به استقرار پروژه انجام شد تا خروجی نمایش و قابل استفاده باشد مسئول **حمیدرضا کناری زاده** است و در اخر فایل پیوست پروژه انجام شد.

چالش ها

۱. فشرده بودن تایم های کاری هر نفر اعضا گروه باعث کند پیش رفتن پروژه شد
۲. مشکل در پیدا سازی در قسمت وب ویو نوار کناری vs code
۳. زمان برای آموزش
۴. باگ هایی درهر قسمت باید رفع می کردیم به علت نبود تجربه لازم