Capstone Project - Vision

Daniel Harder

CST-451 – Sprint 4

Grand Canyon University

Instructor: Professor Donna Jackson

Date: 01/14/2024

## **Project Status**

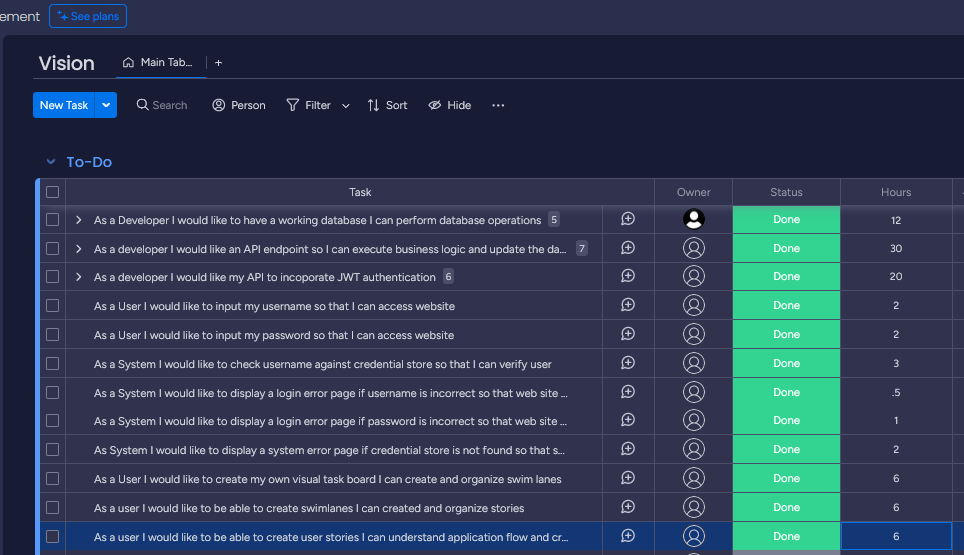
During its development throughout the duration of the project Vision has developed extensively. Database access with Entity Framework, Authentication, CRUD operations for Users, Task Boards, Lanes, Stories, and Tasks were all created. Database design for member permission were also incorporated in the API.

Despite best efforts, Vision still has many features that stand to be implemented before minimum viable project could be reached for users. Given the rich API implementation, Vision’s front-end Angular application would be very likely to utilize the APIs features within two more sprints.

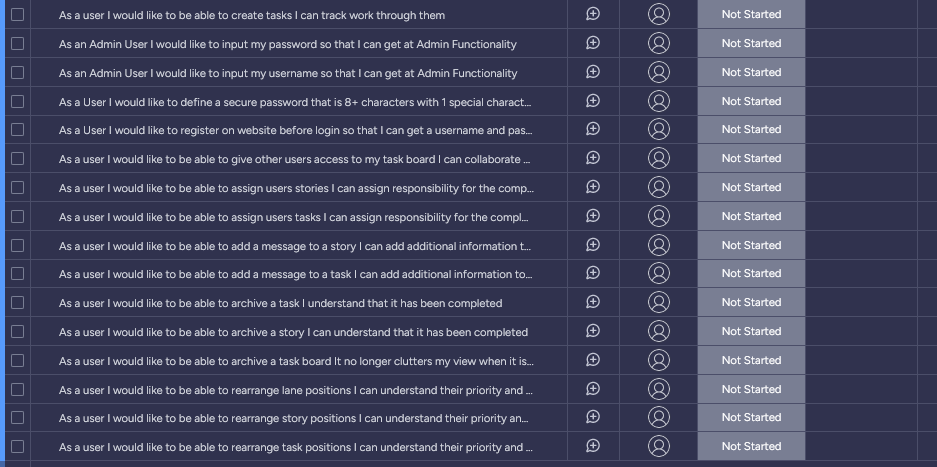
To compare to a real-world scenario, at this stage of the app I would speak to the client about extending the deadline and review the terms of the contract for an extension. In an academic scenario, this is not possible as the deadline for the class has been reached. That said, Vision shows extreme promise for future sprints that will occur outside of the class.

## **Implementation and Requirements**

In the planning stages of the projects the functional requirements were presented in an excel-based matrix. Here we laid out the foundation for our functional requirements. The previously created functional requirements have now been migrated to a work board on Monday.com. Monday.com is an excellent choice for task management as it makes it easy to work with tasks, add time, and add subtasks as needed when staying Agile. This is how task management has been managed for the duration of the project.

*Sprint 4 - 1/15/2023*

*Backlog*



## **Source Code**

The source code is accessible at <https://github.com/danielharder/Vision/tree/Sprint-4>. A copy of the document you are currently reading has also been added to this GitHub repository to act as an overview and demo of the product to public viewers.

## **Application Demo**

Loom Link to demo:

<https://www.loom.com/share/4e1d79982afd4226be5de7d9739adfc6?sid=d9a77bf2-fa85-436d-9b03-75558e6b9046>

## **Unit and Integration Test Plans**

|  |
| --- |
| Test Tools |
| 1 – Manual Testing Procedures |
| 2 – Xunit (depreciated) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TESTING** | | | | | | | | | | | | | | | | | | | | | | | | |
| **Project:** | Vision (Capstone) | | | | | | | |  | **Name of Developer:** | | | | | | | | | Daniel Harder | | | | | |  | |
|  |  | |  |  | |  |  |  | | | | |  |  |  | | | | | |  |  |  | | |  |
|  |  | |  |  | |  |  |  | | | | |  | **Name of Reviewer:** | | | | | | Daniel Harder | | | | |  |
|  |  | |  |  | |  |  |  | | | | |  |  |  | |  | | | | |  |  |  |
| **Unit Test Checklist** | | | | | | | | | | | | | | | | | | | | | | | | |
| **Test** | **Pass** | | Date | | | | | | | | **Reviewed by Developer** | | | | | **Comments** | | | | | | | | |
| **Functionality** | | | | | | | | | | | | | | | | | | | | | | | | |
| N/A |  | |  | | | | | | | |  | | | | | Manual tests have been chosen for the project. | | | | | | | | |
| **Manual Test Checklist** | | | | | | | | | | | | | | | | | | | | | | | | |
| **Test** | | **Pass** | | | Date | | | | | | | **Reviewed by Developer** | | | | | | **Comments** | | | | | | |
| **Functionality** | | | | | | | | | | | | | | | | | | | | | | | | |
| Verify login is successful from UI and make sure pages are only available after auth. | | No | | | 1/14/2024 | | | | | | | Daniel | | | | | | JWT is added to session storage, but CORS policy has bug. | | | | | | |
| Verify user can create Task Boards at the /taskboards page. | | Yes | | | 1/14/2024 | | | | | | | Daniel | | | | | |  | | | | | | |
| Verify user can add a new Task Board at the /taskboards page. | | Yes | | | 1/14/2024 | | | | | | | Daniel | | | | | |  | | | | | | |
| Verify user can view lanes at /taskboard. | | Yes | | | 1/14/2024 | | | | | | | Daniel | | | | | |  | | | | | | |
| Verify user can add new lanes at /taskboard. | | Yes | | | 1/14/2024 | | | | | | | Daniel | | | | | |  | | | | | | |
| Verify user can delete lanes at /taskboard. | | Yes | | | 1/14/2024 | | | | | | | Daniel | | | | | |  | | | | | | |
| Verify user can view stories at /taskboard. | | Yes | | | 1/14/2024 | | | | | | | Daniel | | | | | |  | | | | | | |
| Verify user can add stories at /taskboard. | | Yes | | | 1/14/2024 | | | | | | | Daniel | | | | | |  | | | | | | |
| Verify user can delete stories at /taskboard. | | Yes | | | 1/14/2024 | | | | | | | Daniel | | | | | |  | | | | | | |
| Verify user can logout. | | No | | | 1/14/2024 | | | | | | | Daniel | | | | | |  | | | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ACCEPTANCE TEST FINAL REPORT** | | | | |
| **Project Name: Vision** | | | **Date: 1/14/2024** | |
| **Unresolved Defects** | | | | |
| **Issue/Defect** | **Impact**  **(H, M, L)** | **Risk Mitigation (If known)** | | **Work Around**  **(If known)** |
| **Login is not currently authenticating.** | **H** | **Fix CORS policy before deploying to production.** | | **Do not allow sensitive info to be used for application until authentication is working.** |

## **Document History**

With Sprint 4, the Angular code was added written for the project. This incorporated login, task board creation, lane creation and deletion, and story creation and deletion functionality. There were some minor changes to the REST API, in the form of methods to grab entities that match the have an identifier matching the primary key of their parent entity.

## **Project Planning Matrix**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Task | Dependencies | Status | Effort Hours | Cost | Start Date | Planned Completion | Estimate to Completion |
| 1 | Research the design pattern of the C# / Angular template in Visual Studio. | Visual Studio, .Net Framework, Angular | Not yet started | 4 | NA | 11/20/2023 | 11/25/2023 | 11/25/2023 |
| 2 | Design UML diagrams for application to understand logic that will be utilized. | Draw.io, Windows Notepad | Not yet started | 4 | NA | 11/20/2023 | 11/20/2023 | 11/20/2023 |
| 3 | Design Entity Relationship Diagram to identify what tables and fields need to be set in SQL database. | Draw.io, Windows Notepad | Not yet started | 4 | NA | 11/20/2023 | 11/20/2023 | 11/20/2023 |
| 4 | Utilize Angular documentation to plan UI features. | Angular documentation | Not yet started | 3 | NA | 11/20/2023 | 11/20/2023 | 11/20/2023 |

## **Functional Requirements Matrix**

Below is the functional requirements Matrix. While not all features were implemented in the UI, the majority of the features are possible using the REST API.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Feature** | **As a(n) <actor>** | **I would like to <description>** | **So that <outcome>** |
| F-001 | Registration | As a User | I would like to register on website before login | so that I can get a username and password |
| F-002 | Register | As a User | I would like to define a secure password that is 8+ characters with 1 special character | so that I have strong passwords |
| F-003 |  |  |  |  |
| F-004 | Login | As a User | I would like to input my username | so that I can access website |
| F-005 |  | As a User | I would like to input my password | so that I can access website |
| F-006 |  | As a System | I would like to check username against credential store | so that I can verify user |
| F-007 |  | As an Admin User | I would like to input my username | so that I can get at Admin Functionality |
| F-008 |  | As an Admin User | I would like to input my password | so that I can get at Admin Functionality |
| F-009 |  | As a System | I would like to display a login error page if username is incorrect | so that web site is not accessible |
| F-010 |  | As a System | I would like to display a login error page if password is incorrect | so that web site is not accessible |
| F-011 |  | As System | I would like to display a system error page if credential store is not found | so that support can debug a system issue |
| F-012 |  |  |  |  |
| F-013 | Taskboard | As a User | I would like to create my own visual task board | I can create and organize swim lanes |
| F-014 |  | As a user | I would like to be able to create swimlanes | I can created and organize stories |
| F-015 |  | As a user | I would like to be able to create user stories | I can understand application flow and create tasks |
| F-016 |  | As a user | I would like to be able to create tasks | I can track work through them |
| F-017 |  | As a user | I would like to be able to give other users access to my task board | I can collaborate with them on task management |
| F-018 |  | As a user | I would like to be able to assign users stories | I can assign responsibility for the completion of the story |
| F-019 |  | As a user | I would like to be able to assign users tasks | I can assign responsibility for the completion of the task |
| F-020 |  | As a user | I would like to be able to add a message to a story | I can add additional information to the story |
| F-021 |  | As a user | I would like to be able to add a message to a task | I can add additional information to the task |
| F-022 |  | As a user | I would like to be able to archive a task | I understand that it has been completed |
| F-023 |  | As a user | I would like to be able to archive a story | I can understand that it has been completed |
| F-024 |  | As a user | I would like to be able to archive a task board | It no longer clutters my view when it is no longer needed |
| F-025 |  | As a user | I would like to be able to rearrange lane positions | I can understand their priority and status |
| F-026 |  | As a user | I would like to be able to rearrange story positions | I can understand their priority and status |
| F-027 |  | As a user | I would like to be able to rearrange task positions | I can understand their priority and status |

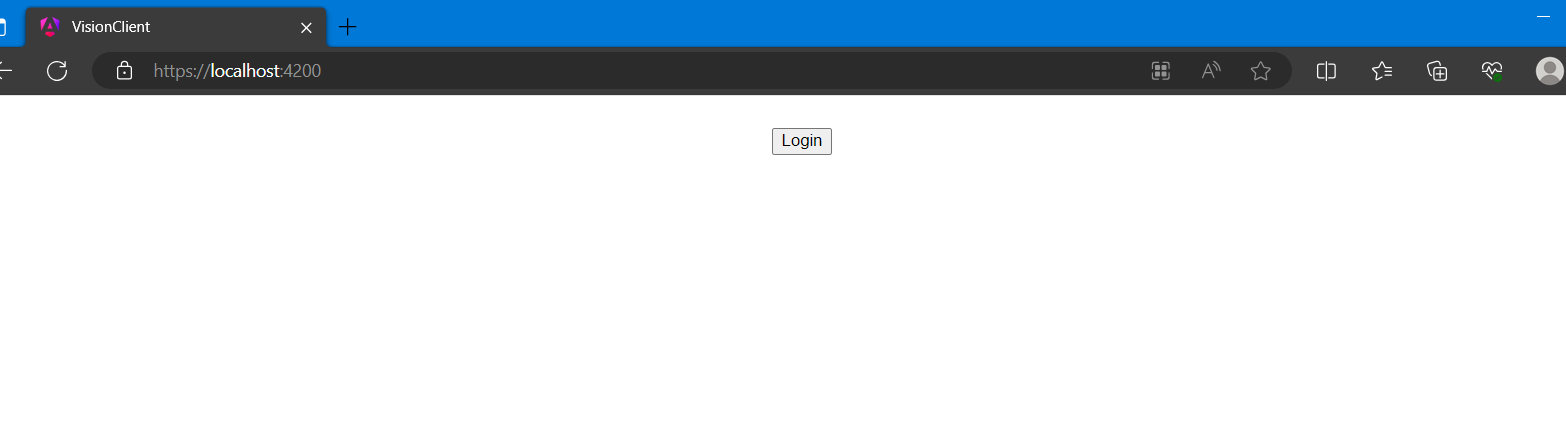
## **User Guide**

**Introduction**

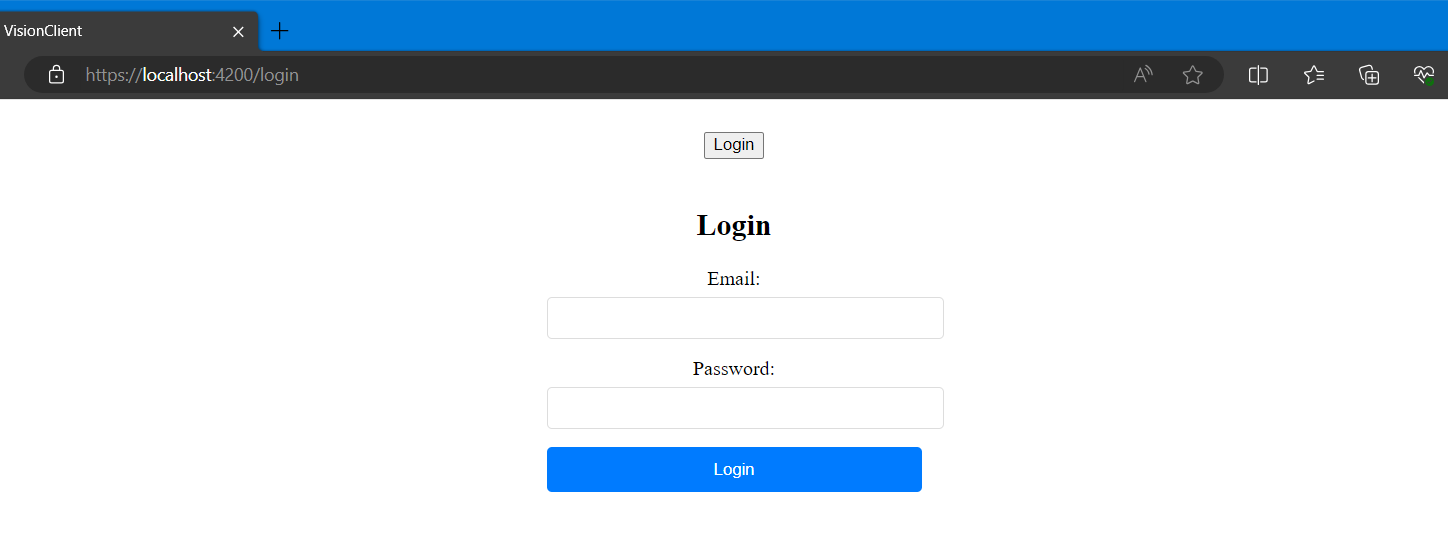
This user guide will give a basic flow for the purpose and function of the website. The site is design to be intuitive to understand, using standard UI elements for navigation. The guide will start with an introduction to the login process, followed by display of the task board and how to use it.

**Overview**

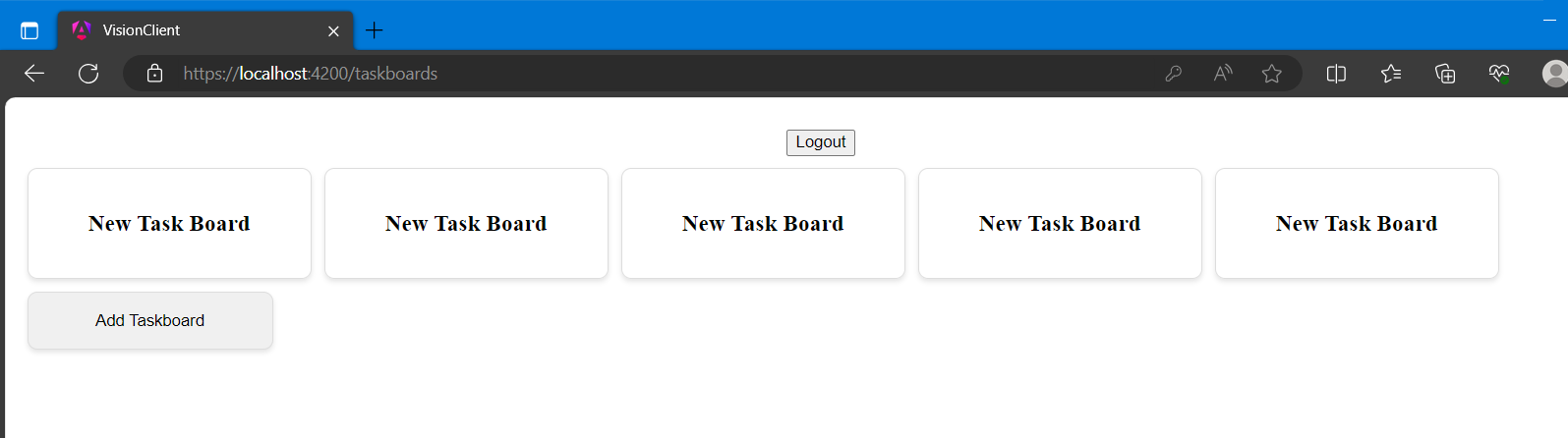
Upon visiting the website the user is presented with a blank page with a login button. Registration is currently an API-only feature and will be added to the website UI at a later time.



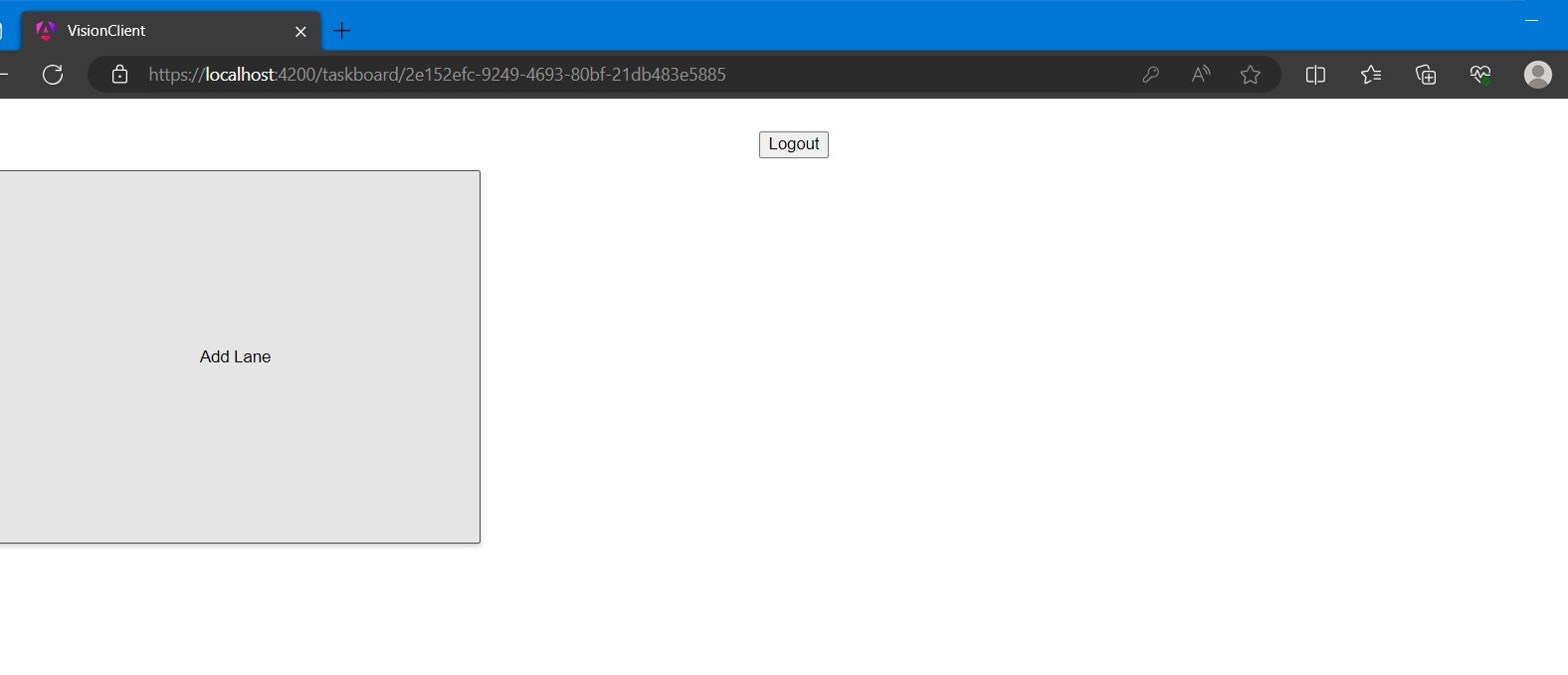
Once the login button is clicked, the user is presented with the option to enter an email and password.

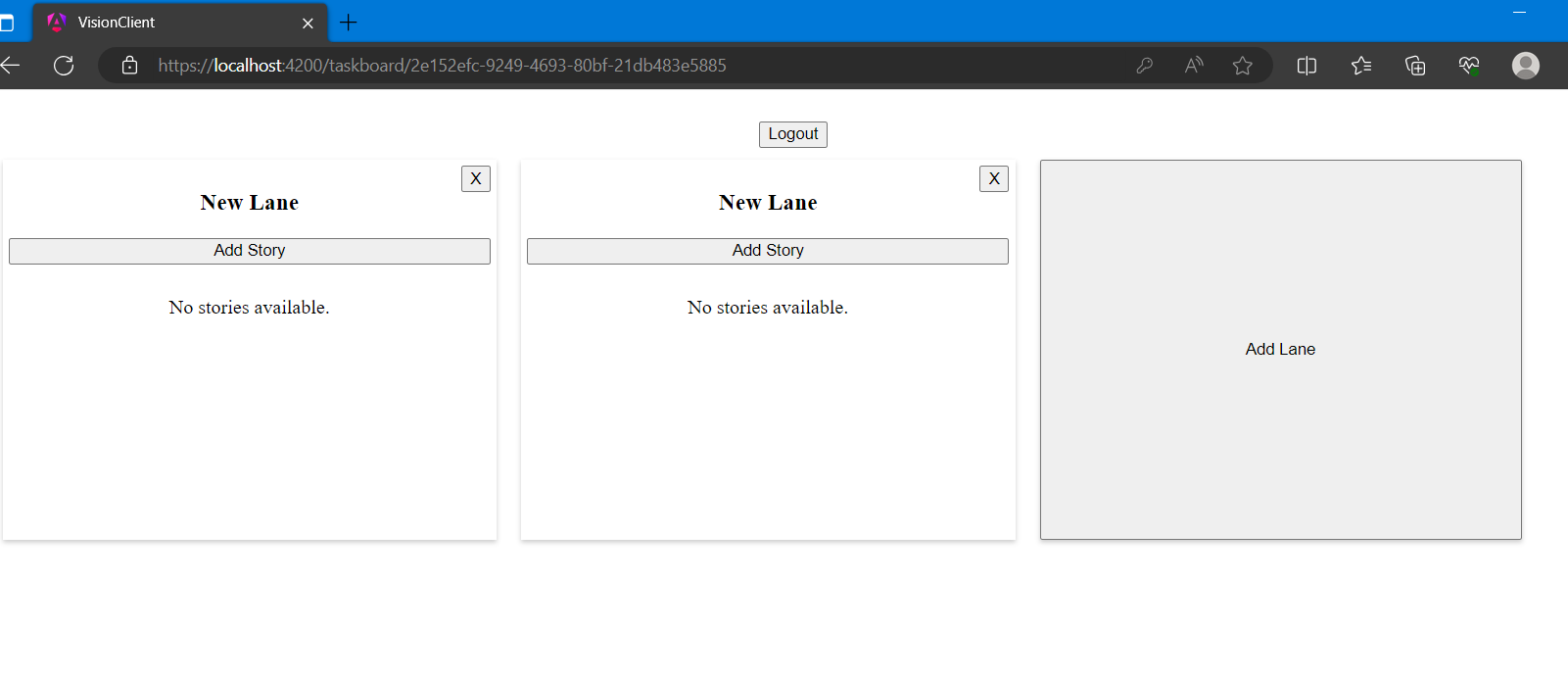


After login, the user will be directed to the TaskBoards page. Here they will be able to see and add new Task Boards using the “Add Taskboard” button. A user can click on a Task Board to open the Task Board view.

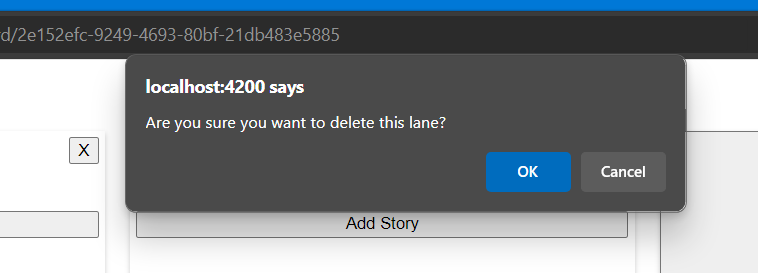
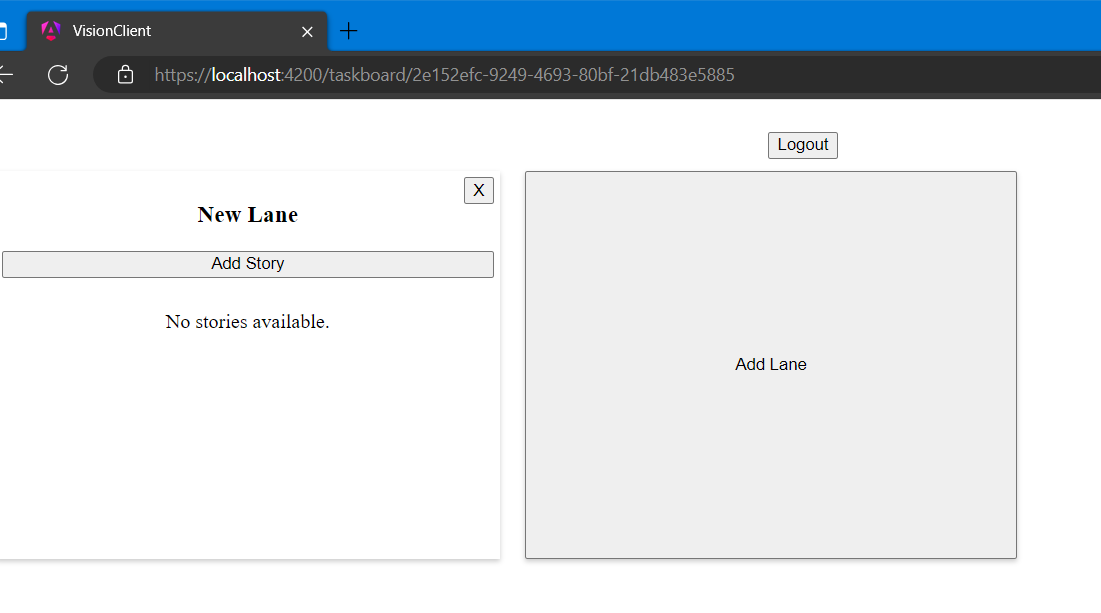


The Task Board view is initially very basic as it only has the option to click “Add Lane”.

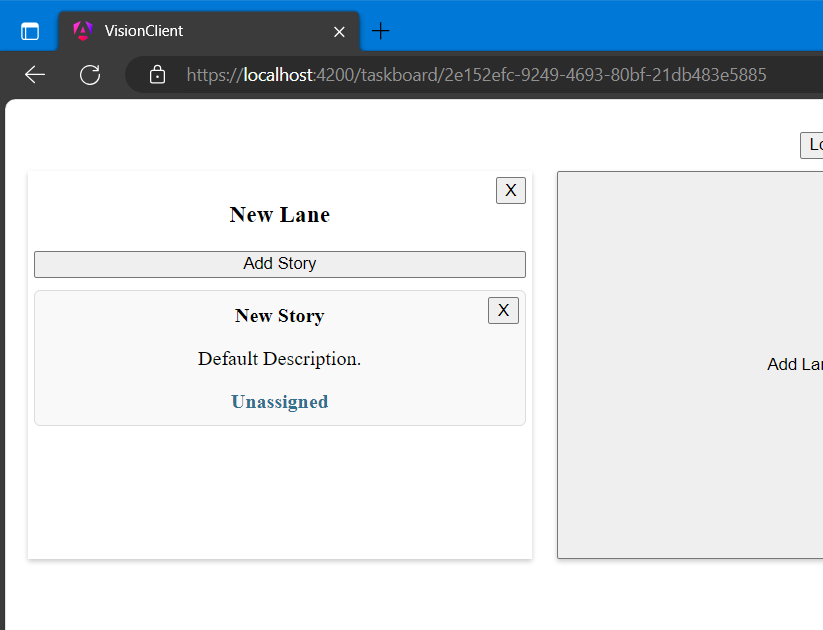


When “Add Lane” has been clicked, 

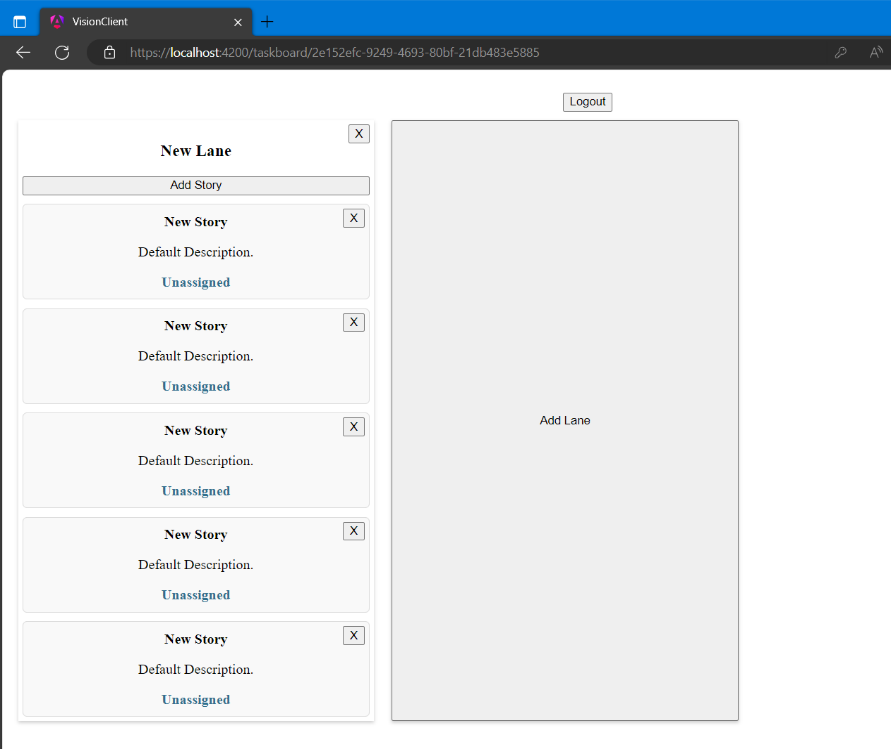
When clicking the “X” in the top right of the lane, the user is asked if they want to delete the lane. If they click “OK” the lane is deleted.

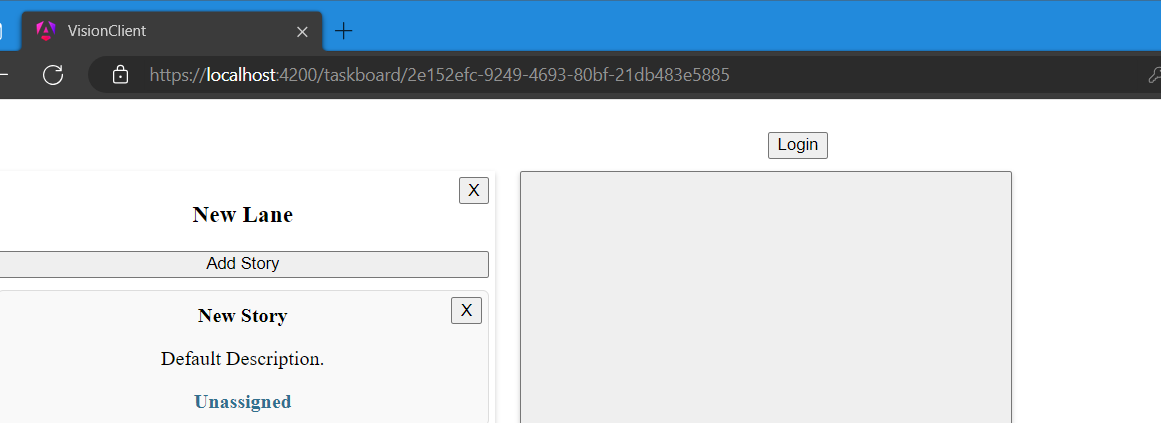
When clicking on “Add Story” a new story is created. The story displays the title, description, and assigned state. These options will be editable in the future, but currently are not. The “X” at the top right of a story can be used to delete the story. There is no prompt to double check that you want to delete a story like there is for a lane.



The user can add as many stories as they want.



Upon clicking “Logout”, the Task Board is still displayed, however the login button shows. Login authentication is functioning, however there is currently a bug preventing validation. Validation and enhanced user permissions will be added in the future, as well as the ability to edit Task Boards, Lanes, and Stories. Stories will be clickable. Upon being clicked they will show a modal that gives the user the ability to add tasks to a story.



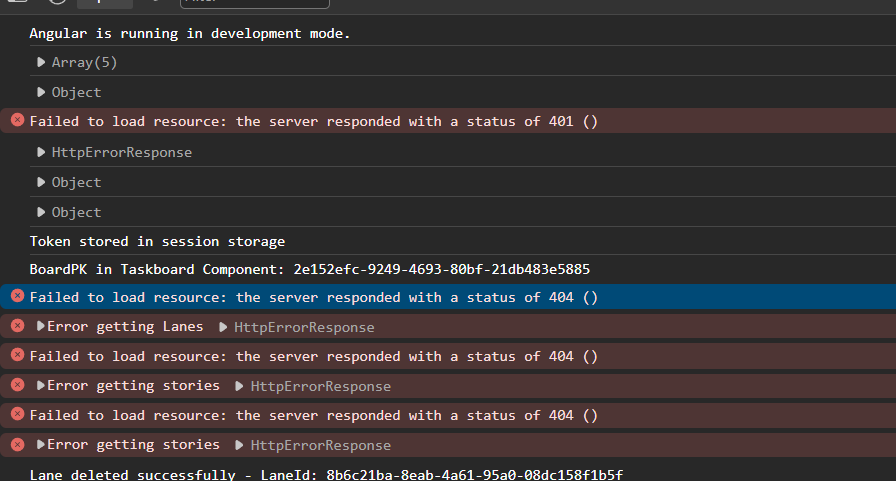
## **Admin Guide**

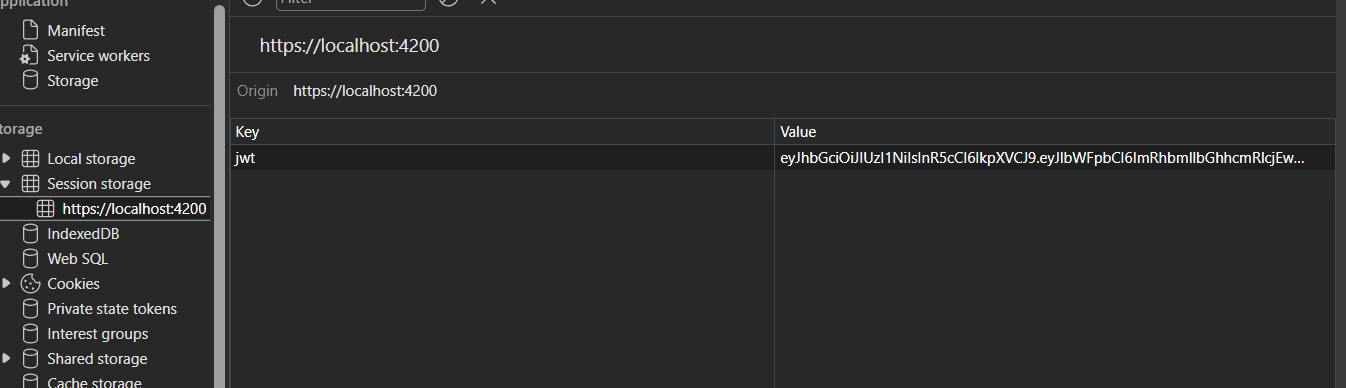
**Introduction**

This admin guide will showcase the current bugs that admins should know about. It will also review information that an admin will need to know when deploying Vision.

**Known Issues – Login**

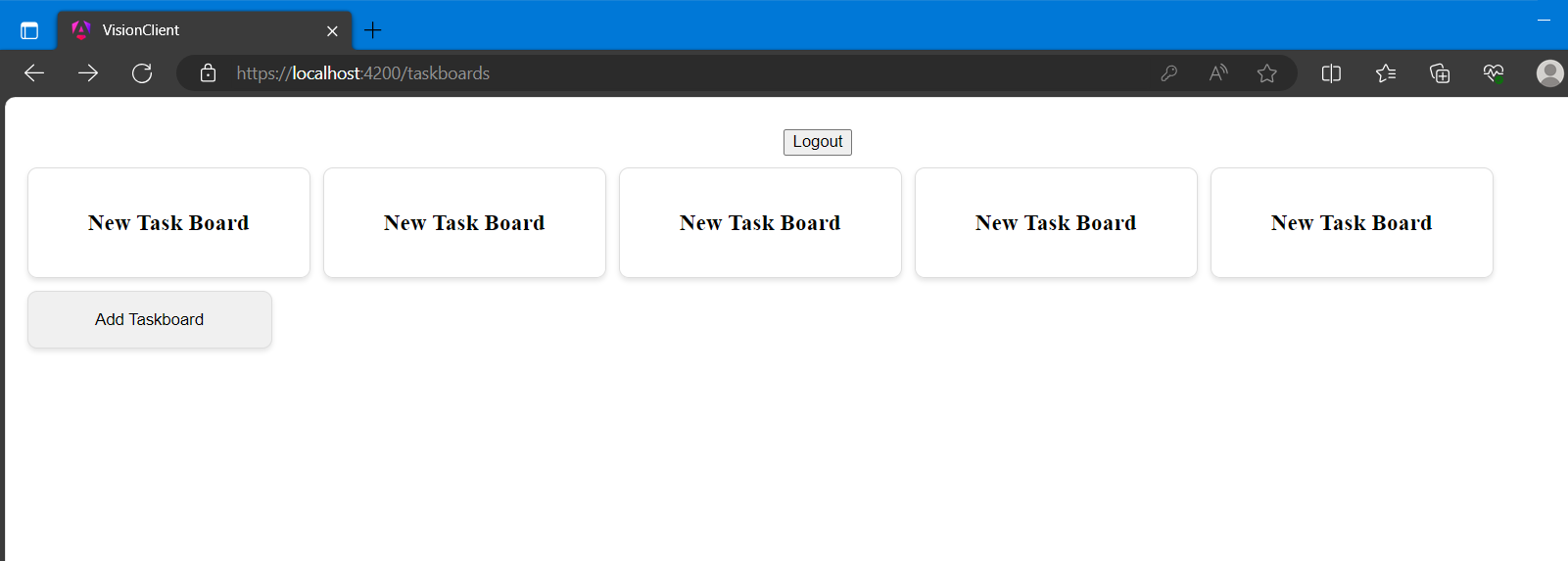
In its current state, ASP.net Web API is not requiring authentication to access the end points. This is due to setting configurations with the CORS policy on the app. This bug will be alleviated in a future release. Login does have some functionality however, and understanding how it works will help administrators understand the structure when the CORS bug is alleviated.

When a login fails, the console currently shows an HTTPS 404 error response. This means that the user failed to authenticate. Likewise, a success message will send a message to the console.

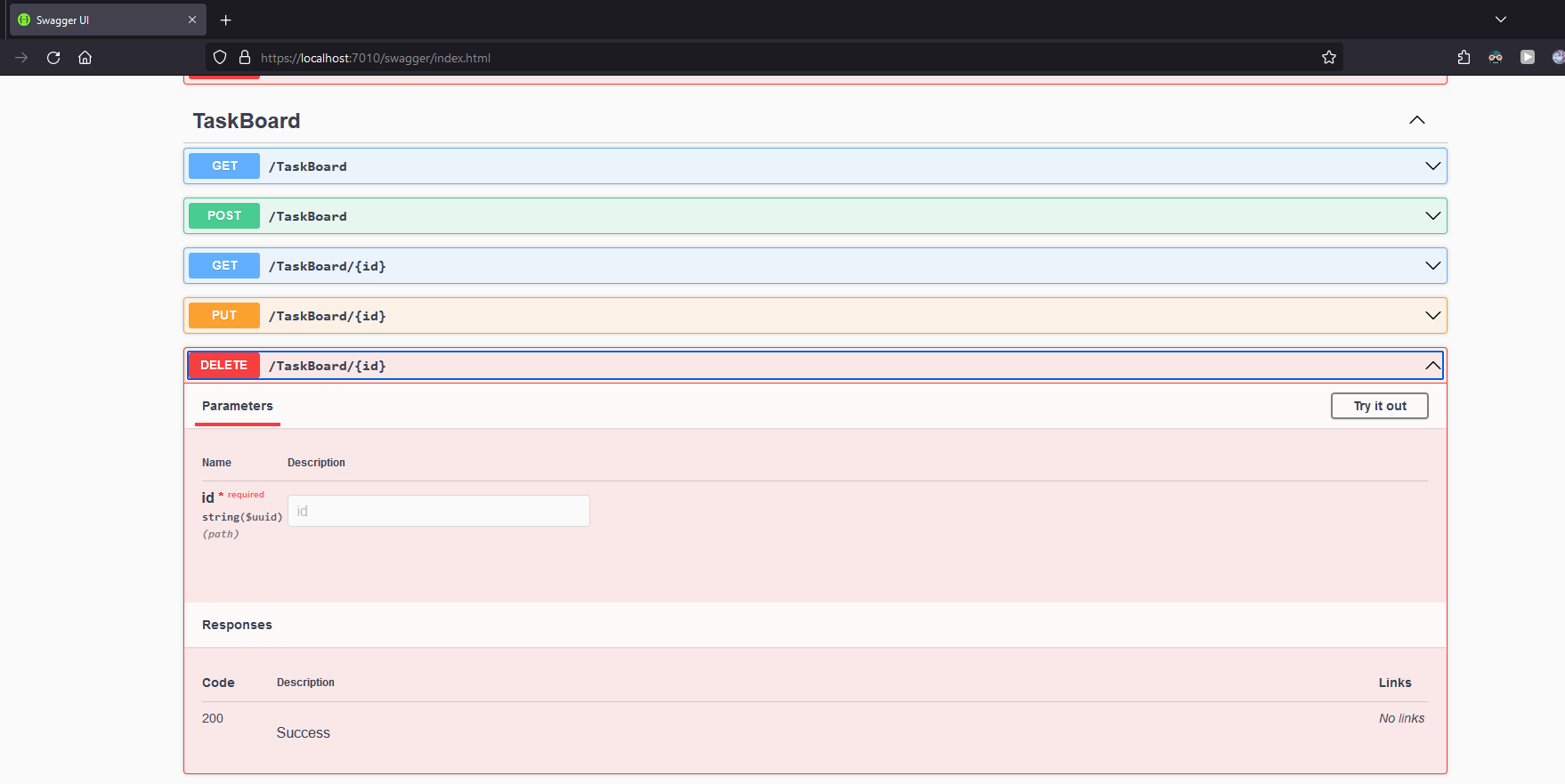
When a user successfully logs in, a JSON Web Token is stored in the browser’s session storage. The key is ‘jwt’ and the value is the token that will be used for auth. When the CORS bug is fixed, this token will be used by Angular to authenticate with the API. Without it, loading pages should result in error messages. 

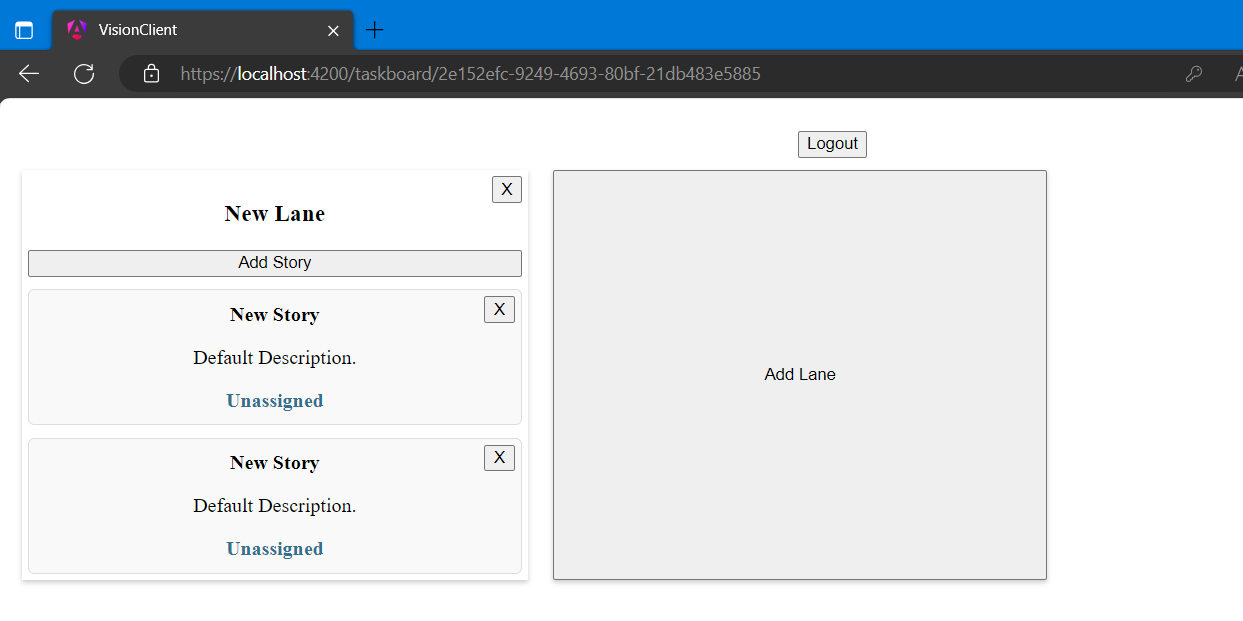
**User Support**

Administrators may need to support users as they user the app. The app currently has limited functionality, which may confuse users. Currently Task Boards cannot be removed or edited from the UI. They can only be added.

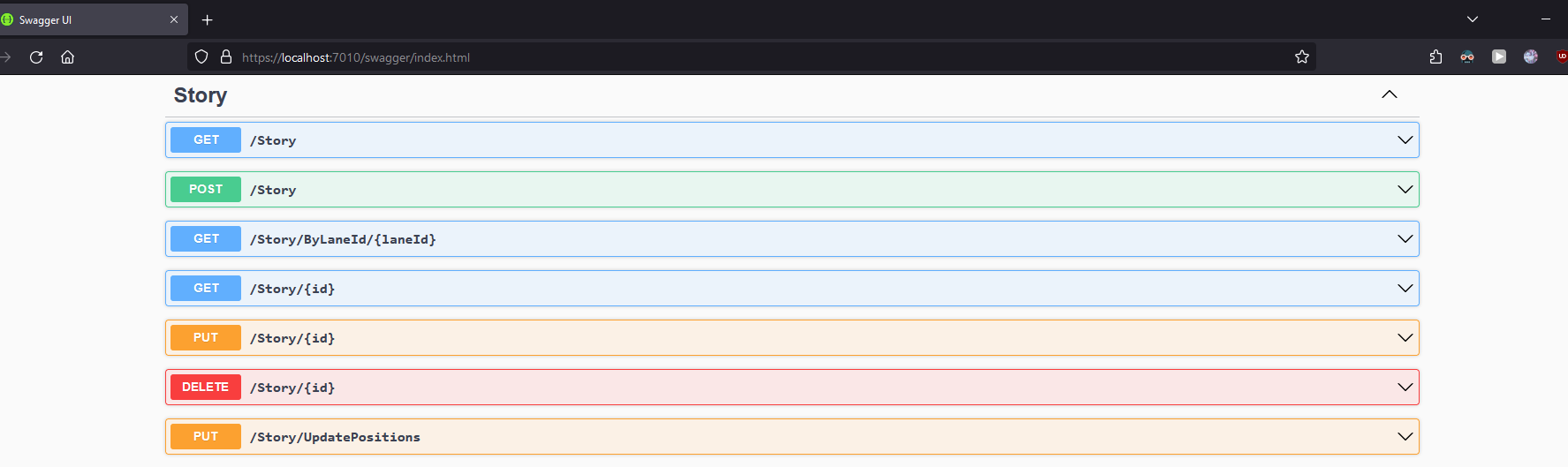


If a user must edit or delete a Task Board, they can do so by sending a REST request. Here is a demonstration of a delete request using Swagger UI. This can also be done with an application like postman.



Once a Task Board has been clicked, the user enters the Task Board view. Here the user can create and delete lanes and stories.

In the current iteration, users cannot edit stories. In the future thy will simply be able to click on the text for one of the fields, converting it to an editable text box. Once they edit the text in this box, they will be able to hit enter to update the field. For now, the fields will need to be edited using REST API calls, like with the Task Board.



**App Deployment**

In addition to supporting users, administrators may need to deploy the app to new web servers. When doing this it is important to keep the architecture in mind. Vision is designed to have a back-end server that hosts the rest API, performs the business logic, and performs database operations. It also has the front-end JavaScript UI. These will need to be hosted on different ports, if not on different servers entirely.

To configure the app, an administrator will need to define the Issue (the server hosting the API) and the Audience (The server which is allowed access to the API) within the appsettings.json file. They will also want to use a different JSON web token key than the key used for the test environment. Please keep in mind The exact details of this deployment will vary depending on the hosting solution chosen.

