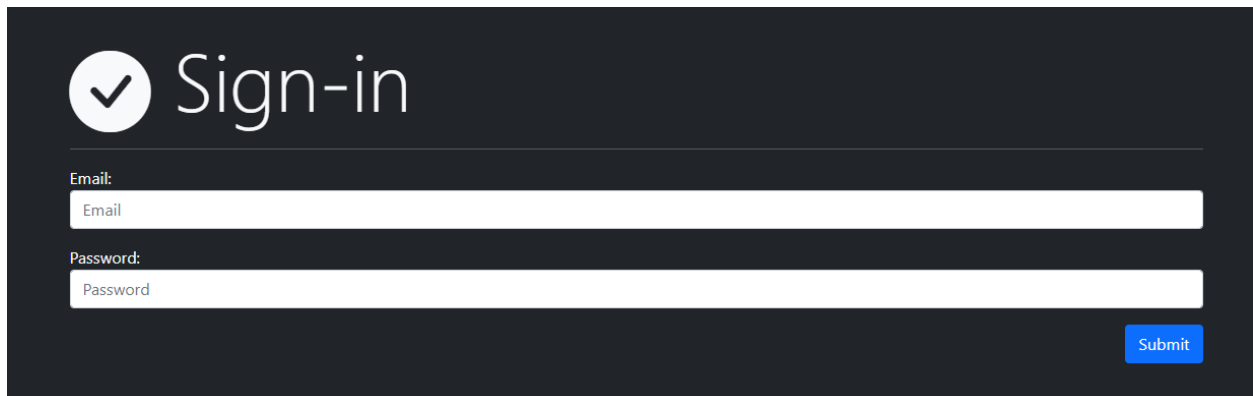


Login View :



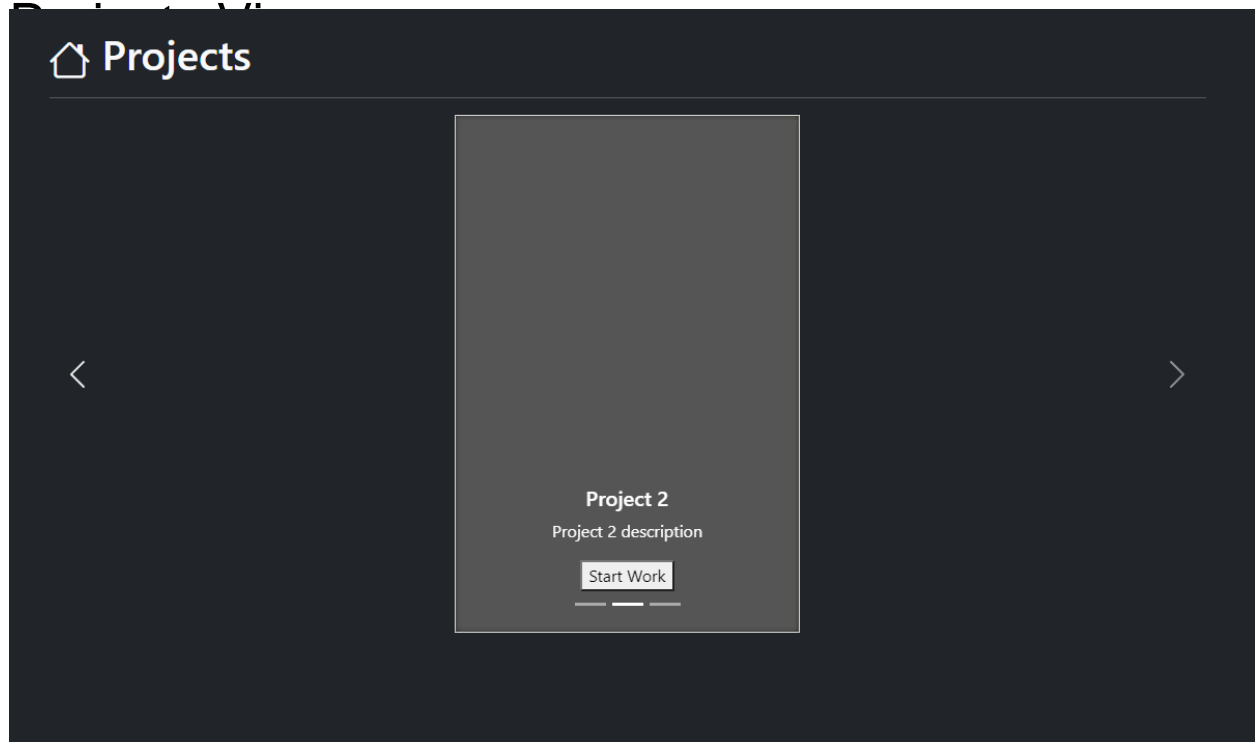
The mockup shows a dark-themed login interface. At the top left is a circular icon containing a white checkmark. To its right is the text 'Sign-in' in a large, white, sans-serif font. Below this, there are two white input fields. The first is labeled 'Email:' and the second is labeled 'Password:'. Both labels are in a small, white, sans-serif font. To the right of the 'Password:' field is a blue button with the word 'Submit' in white. The entire form is set against a dark, solid background.

Expected Executions:

- Functions:
 - Authenticate(): Call function to authenticate user
 - Create(): Create new user (Controller)
 - SubmitButton(): Redirect to Projects view (Controller)
 - ForgotPassword(): Prompt user's email and send an email containing user's password if an account exists with that email. (Controller)
- Server functions:
 - Get User info:
 - Put User info:
- Libraries Needed:
 - HTML, CSS, Bootstrap, React*, JSON, JS

Description:

Login page to authenticate user information or create account if user has no account. Should throw error if user tries to create account with an email already in use.



Expected Executions:

Functions:

Authenticate(): determine if a user has read-only or read/write access to a project

StartWorkButton():Redirect to Task view (Controller)

AddNewProject():Create a new project and give read/write permission to user who created it (Controller)

Server Functions:

GET basic Project data

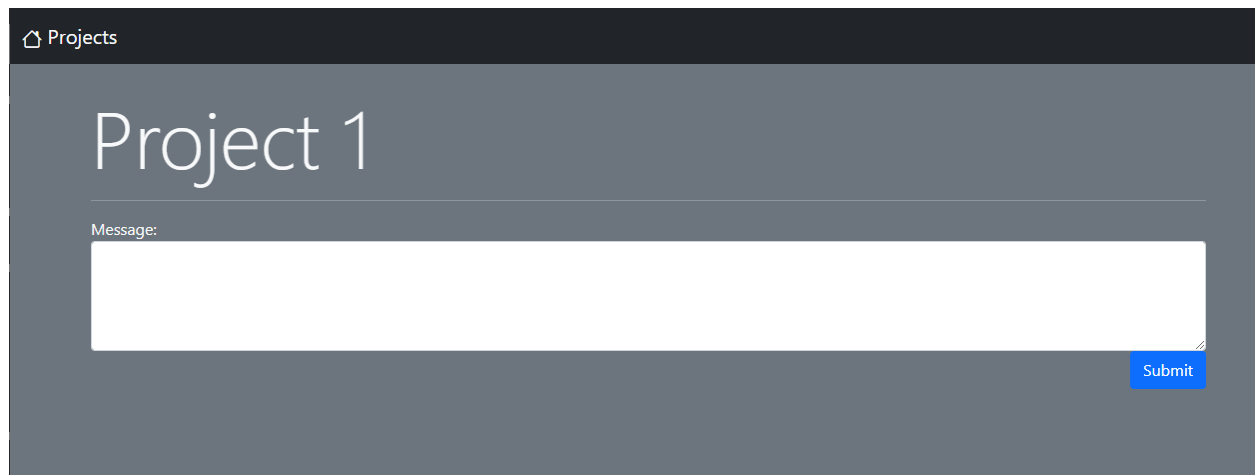
Libraries Needed:

- HTML, CSS, Bootstrap, React*, JSON, JS

Description:

List of current projects for user to choose from. Should display basic information about each project.

Task View:



The screenshot shows a web application interface. At the top, there is a dark header bar with a home icon and the text 'Projects'. Below this, the main content area has a dark background. The title 'Project 1' is displayed in large white text. Underneath the title, there is a label 'Message:' followed by a large white text input field. To the right of the input field, there is a blue button with the text 'Submit'.

Expected Executions:

Functions:

- SubmitButton():Add content to project/task (Controller)
- AddUser():Allow user to edit project/task data (Controller)
- AddNewTask():Create new task within the current project (Controller)

Server Functions:

- GET detailed Project data and task data
- PUT new data as added by user

Libraries Needed:

- HTML, CSS, Bootstrap, React*, JSON, JS

Description:

Retrieves all tasks designated to specific project, and displays them in this view. Expectation is that, Request to server will be made, where task information is stored. Rather than making a new html page for each project, a template will be used, and filled with JS/REACT code to display individual project.

Task View(Inner):

Task X		
Due: XX/XX/XX	Provide reports on ____.	Members:
Assigned by:		- jim
x	Reports should be in x format	
	Example: https:Link	
Files:		
<input type="text"/>	<input type="text"/>	
<input type="text"/>	<input type="text"/>	
<input type="text"/>	<input type="text"/>	

Expect Executions:

Functions:

Attach(): Attaches link to File associated with task, turn in deliverable

Delete(): Removes task from project, and returns to task view

Adjust(): Adjust details of task

display(): Display's task details, including associated file links, Due Date, Expected return, members assigned

view(): View deliverables without adjusting (Viewing a task that you do not have edit rights to)

Server Functions:

GET specific details of task (Due date, File links, members, etc.)

PUT file for deliverable

Backend:

Authenticate user: ensure permissions for tasks are distributed correctly I.E: Ability to adjust/delete task, or ability to turn in/view deliverable

Description:

The innermost view of a project: Viewing a specific task to be completed, and all its parameters. This is where the associated files are linked, and where all the details of specific tasks are stored. This page should allow users to drop in their files for delivery, as well as view any associated files. Project Manager/admin should be able to view and adjust task details as needed, Users the task is assigned to should be able to turn in deliverables, all others should not see the task if not assigned, or be able to GET files but not put.