<nodebucket>

TECHNICAL DESIGN DOCUMENT

|  |  |
| --- | --- |
| **Author** | *Shane Hingtgen* |
| **Last Updated** | 8/31/2023 8:13 PM |
| **Version Number** | *0.9.0* |

**Version History**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Date | Reason for Change | Version |
| Shane Hingtgen | 8/7/23 | Created Initial TDD & Purpose | 0.1.0 |
| Shane Hingtgen | 8/10/23 | Added Terminology & User stories | 0.2.0 |
| Shane Hingtgen | 8/11/23 | Added prototype and site map | 0.3.0 |
| Shane Hingtgen | 8/15/23 | Added ORD & NoSQL Data structure | 0.4.0 |
| Shane Hingtgen | 8/16/23 | Updated Terminology | 0.4.1 |
| Shane Hingtgen | 8/17/23 | Added Function and unit Tests | 0.5.0 |
| Shane Hingtgen | 8/18/23 | Added Reqs and Swagger documentation | 0.6.5 |
| Shane Hingtgen | 8/22/23 | Add Unit test, functional test and swagger documentation | 0.7.0 |
| Shane Hingtgen | 8/24/23 | Corrected the soapUI and swagger docs | 0.7.5 |
| Shane Hingtgen | 8/29/23 | Fixed reqs and added swagger documentation | 0.8.3 |
| Shane Hingtgen | 8/31/23 | Adding unit tests and swagger docs | 0.9.0 |

Table of Contents

[SECTION 1: INTRODUCTION 3](#_Toc142489796)

[1.1. Purpose 3](#_Toc142489797)

[1.2. Terminology 3](#_Toc142489798)

[1.3. User Personas 3](#_Toc142489799)

[1.4. User Stories 3](#_Toc142489800)

[1.5. Work Estimations 3](#_Toc142489801)

[SECTION 2: PROCESS DESIGN 3](#_Toc142489802)

[2.1. Sitemap 3](#_Toc142489803)

[2.2. Prototypes 3](#_Toc142489804)

[2.3. ORD (Object Relational Diagram(s) 3](#_Toc142489805)

[2.4. NoSQL Data Structure 3](#_Toc142489806)

[SECTION 3: FUNCTIONAL DESIGN 4](#_Toc142489807)

[3.1. Overview of the Design 4](#_Toc142489808)

[3.2. Functional Design 4](#_Toc142489809)

[3.3. Feedback/Comments 4](#_Toc142489810)

[SECTION 4: QA TESTING 4](#_Toc142489811)

[4.1. QA Test Plan 4](#_Toc142489812)

[4.2. Unit Tests 5](#_Toc142489813)

[SECTION 4: REFERENCES 5](#_Toc142489814)

# SECTION 1: INTRODUCTION

## Purpose

The purpose of this application is to with a secure way to manage tasks. Tasks should have two columns, ToDo and Complete with multiple categories to organize. It must have a log in page. In addition, the application should have a home page, about us page, contact us page so users know what the company is about and has a way to contact if needed.

## Terminology

|  |  |
| --- | --- |
| **Name** | **Comments** |
| *MEAN Stack* | *MongoDB, Express, Angular, and Node.js* |
| *VS Code* | *Development IDE* |
| *SoapUI* | *API testing tool* |
| *MongoDB Compass* | *Database development IDE https://www.mongodb.com/* |
| *Angular Material* | [*https://material.angular.io/*](https://material.angular.io/) |
| *Flex-layout* | [*https://github.com/angular/flex-layout*](https://github.com/angular/flex-layout) |
| *Node.js* | https://nodejs.org/en/about |
| *Express* | https://expressjs.com/ |
| *Angular Material* | https://material.angular.io/ |
| *Swagger* | API Documentation and design tool https://swagger.io/ |

## User Personas

# Persona 1 - Dan Patel



(Pexels, 2020)

## Age: 43

## Gender: M

## Job Title: Software Developer

## Marital Status: M

**Characteristics**: healthy, wears glasses because he is blind without them, loves gaming with his kids, very knowledgeable programmer

## Short Bio: Dan is an experienced software developer, having been working in the industry for almost 20 years. He is married with children. Dan enjoys spending time with his family and going on vacations to foreign countries. Very computer savvy, knows computers very well, and in his free time attempts to make games with unreal engine as a hobby for his kids.

**Feature Most likely to use:** Will use dev features for creation of tasks, will even use tasks and sign in

**Feature to Not likely to use:** home, contact, or about

**Application Needs:** He needs VS code, node, angular, express, mongoDB

**Computer Skill:** Advanced

**Hobbies:** Creating Video Games, Trips to different countries

**Future Vacations:** Greece

**Social Media Accounts:**

Facebook, Twitter, LinkedIn

# Persona 2 - Steven Prior



(Miroshnichenko, 2020)

## Age: 35

## Gender: M

## Job Title: Marketing Manager

**Characteristics:** Athletic, glory days was his college football days playing for Bellevue University, very motivated

## Marital Status: Single

## Short Bio: Steven is a Marketing Manager for a large engineering firm. He enjoys marketing many products for his employer. He has been in this management position for almost 3 years now. He is single and has no kids. He loves sports and sports betting. He is a very organized and hardworking individual.

**Feature Most likely to use:** Would like to see about us to determine if working with this company is a good investment and a contact page. Also, a tool for tracking tasks

**Feature to Not likely to use:** Likely to use the application to leverage his work position even further.

**Application Needs:** Task management, log in, about us, and contact us, needs phone numbers.

**Computer Skill:** Mid-High

**Hobbies:** Watching sports, especially football and soccer, likes to play ultimate frisbee and golf

**Future Vacations:** Italy

**Social Media Accounts:** Facebook, Twitter, LinkedIn, TikTok, Instagram

# Persona 3 - Miriam Nuero



(A. Piacquadio, 2020)

## Age: 50

## Gender: F

## Job Title: Project Manager

**Characteristics:** Ready to retire, content with work position, loves her job but happy to be off work. Looking forward for vacations with this newfound life of no kids in the house. Loves her kids and is proud of them. Very organized individual which relates to her job position.

## Marital Status: Widow

## Short Bio: Miriam Nuero is a recent widow when her husband passed away last year. Miriam has 5 kids and is not only a dedicated mother but enjoys her position as a project manager at a mid-sized engineering firm. She manages projects for the field employees and works remotely. She only uses computers at work and currently manages her projects through excel. Would like an easy way to manage tasks. Probably is using her computer much besides work and Facebook. Avid reader.

**Feature Most likely to use:** most likely to use tasks to help with project management

**Feature to Not likely to use:** probably doesn’t care about the about us page or contact page unless the feature is broken, gets work done and closes laptop

**Application Needs:** Tasks manager, log in

**Computer Skill:** mid level, not really looking to learn more computer skills

**Hobbies:** Reading romance novels, sitting at the beach, and travelling to visit her kids

**Future Vacations:** Mexico

**Social Media Accounts:** Facebook to keep track of her kids

## User Stories

|  |
| --- |
| Dan Patel  **Story** |
| As a technical user I need API documentation for task endpoints So I can integrate task management into our custom tools |
| As a technical user I want a CLI for quick task creation To streamline adding tasks without leaving the terminal |
| As a technical user I need to categorize tasks by project So I can easily filter tasks related to specific projects |
| As a technical user I need webhook support for task updates To receive real-time notifications in our development chat |
| As a technical user I want keyboard shortcuts for task actions To navigate and manage tasks more efficiently |

|  |
| --- |
| Steven Prior  **Story** |
| As a user I need task prioritization To focus on high-priority tasks and meet campaign deadlines |
| As a user, I want to be able to separate tasks by client to be able to keep them organized |
| As a working professional, I need to be able to work on mobile due to my busy schedule |
| As a marketing manager, I need to plan and execute a social media campaign to promote our upcoming product launch. |
| As a user I need a high level of security so my clients data is protected |

|  |
| --- |
| Miriam Nuero  **Story** |
| As a user, I want access secure in order to honor NDAs I have signed with clients |
| As a project manager, I need to create tasks to lead kickoff meetings for a new client project. |
| As a pm I need a way to change project status to complete so I can keep track of what’s been complete |
| As a Project Manager, I want to efficiently assign tasks to team members and track their progress to ensure the project stays on schedule and meets its objectives. |
| As a pm I need to have a way to prioritize tasks in order so I can see which deadlines are coming up |

## Work Estimations

|  |  |  |
| --- | --- | --- |
| **Story 1** | **Points** | **Work Hours** |
| As a technical user I need API documentation for task endpoints So I can integrate task management into our custom tools   * Create a POST API **Work Hours**:1 * Create a GET API to recall the tasks **Work Hours**: 1 * Create a DELETE API to remove tasks **Work Hours**: 1 * Create a way to edit tasks **Work Hours**: 1 * API Testing Integrate communication into POST API **Work Hours: 1** * Site integration **Work Hours**: 1 | 3 | 6 |
| As a technical user I want a CLI for quick task creation To streamline adding tasks without leaving the terminal   * Integrate communication into POST API **Work Hours**: 1 * Integrate communication into GET API **Work Hours**: 1 * Integrate communication into GET edit API **Work Hours**: 1 * Integrate communication into DELETE API **Work Hours**: 1 * Create Messages for creation of tasks **Work Hours**: 1 | 1 | 5 |
| As a technical user I need to categorize tasks by project So I can easily filter tasks related to specific projects   * Create a list of categories in todo **Work Hours**: .5 * Create a way to separate Categories in todo **Work Hours**: .5 * Create way to move categories into complete **Work Hours**: 1 * Create a list of categories in Complete **Work Hours**: .5 * Create a way to separate Categories in complete **Work Hours**: .5 | 3 | 3 |
| As a technical user I need webhook support for task updates To receive real-time notifications in our development chat   * Need way to send messages to dev chat for 400 errors **Work Hours**: 3 * Need way to send messages to dev chat for 500 errors **Work Hours**: 3 * Need way to create a way for connection to APIs into teams **Work Hours**: 3 * Need way to create a task for errors **Work Hours**: 2 * Need a separate chat for notifications **Work Hours**: 1 | 5 | 12 |
| As a technical user I want keyboard shortcuts for task actions To navigate and manage tasks more efficiently | 1 | 1 |
| **Story 2** | **Points** | **Work Hours** |
| As a user I need task prioritization To focus on high-priority tasks and meet campaign deadlines   * Add categories to API creation **Work Hours**:1 * Add way to sort by date **Work Hours**: 1 * Add way to sort by category **Work Hours**: 1 * Add way to sort by priority **Work Hours**: 1 * Add way to create priority **Work Hours**: 1 | 1 | 5 |
| As a user, I want to be able to separate tasks by client to be able to keep them organized   * API for Task creation **Work Hours**:1 * API to call for tasks **Work Hours**: 1 * API to edit tasks **Work Hours**: 1 * Add clients to API Creation **Work Hours**: 1 * Add way to filter by client **Work Hours**: 1 | 5 | 5 |
| As a working professional, I need to be able to work on mobile due to my busy schedule   * Create Task page mobile friendly **Work Hours**: 2 * Create sign in page mobile friendly **Work Hours**: 2 * Create home page mobile friendly **Work Hours**: 2 * Create contact us mobile friendly **Work Hours**: 2 * Create about us mobile friendly **Work Hours**: 2 | 3 | 10 |
| As a marketing manager, I need to plan and execute a social media campaign to promote our upcoming product launch.   * Draft campaign strategy **Work Hours**: 1 * Draft campaign ads **Work Hours**: 1 * Find advertisers **Work Hours**: 1 * Appoint a someone to social media associate **Work Hours**: .5 * Launch campaign **Work Hours**: .5 | 3 | 4 |
| As a user I need a high level of security so my clients data is protected   * Encrypt passwords **Work Hours**: 5 * Hashout typed passwords **Work Hours**: 1 * Add two factor authentication **Work Hours**: 3 * Add layer of security to DB **Work Hours**: 2 * Lockout frequent invalid userid or passwords **Work Hours**: 1 | 5 | 12 |
| **Story 3** | **Points** | **Work Hours** |
| As a user, I want access secure in order to honor NDAs I have signed with clients   * Document NDA’s **Work Hours**: 1 * Separate log ins for clients **Work Hours**: 1 * Encrypt clients data **Work Hours**: 6 * Determine NDA requirements **Work Hours**: 1 * Implement requirements **Work Hours**: 6 | 5 | 15 |
| As a project manager, I need to create tasks to lead kickoff meetings for a new client project.   * API for task creation **Work Hours**: 2 * API to edit tasks **Work Hours**: 2 * API for assigning tasks **Work Hours**: 2 * API to call tasks in sort tasks **Work Hours**: 2 * Formal application to present tasks **Work Hours**: 4 | 1 | 12 |
| As a pm I need a way to change project status to complete so I can keep track of what’s been complete   * API to create projects **Work Hours**:2 * Category for project status **Work Hours**: 2 * API to fetch projects by status **Work Hours**: 2 * Filter projects for status **Work Hours**: 2 * Formal UI to view projects **Work Hours**: 4 | 1 | 12 |
| As a Project Manager, I want to efficiently assign tasks to team members and track their progress to ensure the project stays on schedule and meets its objectives.   * API for task creation **Work Hours**: 2 * API to edit tasks **Work Hours**: 2 * API to delete tasks **Work Hours**: 2 * API to assign tasks **Work Hours**: 2 * Filter by dates **Work Hours**: 1 | 3 | 9 |
| As a pm I need to have a way to prioritize tasks in order so I can see which deadlines are coming up   * API to create projects **Work Hours**:2 * Category for project priority **Work Hours**: 2 * API to fetch projects by priority **Work Hours**: 2 * Filter projects for priority **Work Hours**: 2 * API to assign tasks projects **Work Hours**: 2 | 1 | 10 |

Some assumptions made such as mobile development whether it was part of the initial design, It is also difficult to gauge how long or how difficult something is. Often over analyzing the tasks makes it seem more difficult than it probably is. Assumed we will be marking this in a social media campaign, project has NDA’s, mobile design, categories. We will now be implementing some of the ideas such as categories and mobile application. We assumed that we would need notifications to dev chat for real time support. Assuming that we would want keyboard shortcuts, which is another idea being implemented. Assumed we would need CLI for creating tasks. Some of these ideas are good, and if this were a project outside of class would be ideas fit for back log once the initial program has been completed. Due to lack of experience every task may take a bit to implement, may need to find outside resources to assist in password encryption if this application had password and security. Its also difficult to account for an roadblocks developers may have while creating the applications which could further push the amount of hours worked on the application. A lot of the points made during the stories left out the creation of a database, which may lead to a problem during development.

# SECTION 2: PROCESS DESIGN

## 2.1. Sitemap

## A diagram of a company Description automatically generated

## 2.2. Prototypes

[*https://www.eggradients.com/blog/ravenclaw-colors*](https://www.eggradients.com/blog/ravenclaw-colors) *House of RavenClaw Colors*

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a contact us page

Description automatically generatedA screenshot of a computer error

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Some best practices include a simple user interface for creating tasks. Keeping the form inputs short and in line will allow users to understand the form much better than having it on different parts of the page, or even too many form fields could lead to confusion. Short and sweet, makes it easy to use. Added a red trash icon but may also add text next to the Icon so users know where to delete tasks. Category is a drop-down menu and not a radio button, since we will start with five categories which seems like too many radio buttons. Login submits, create account button, and create task are all buttons to draw users’ attention.

## 2.3. ORD (Object Relational Diagram(s)

A close-up of a white background

Description automatically generatedA diagram of a diagram

Description automatically generated

## 2.4. NoSQL Data Structure

A screen shot of a computer code

Description automatically generated

# SECTION 3: FUNCTIONAL DESIGN

## 3.1. Overview of the Design

The theme is House Ravenclaw and applied to all the pages on the application, The home page, contact us page, and about us page are available to both non auth users and auth users. Application must be mobile friendly. Sign in will redirect to the Tasks page, and the sign out will direct to the home page. Application must have the appropriate thrown errors, such as an empId not found 404, or issue with URL not found 404. Sign in must have errors for invalid empId, and must show message to be a numerical value. All error messages, sign in messages, connection messages and close of connection messages should be logged to the console.

## 3.2. Functional Design

|  |  |
| --- | --- |
| **Requirement #** | **Design Definition** |
| R - 1 | Theme: House Ravenclow  Styles to resemble the colors provided in document link: <https://www.eggradients.com/blog/ravenclaw-colors>  Will use a combination of Angular Material, Bootstrap, and CSS |
| R - 2 | Home page  Angular:   * [home.component.html](http://home.component.html) * [home.component.css](http://home.component.css) * [home.component.ts](http://home.component.ts) * base-layout.component.html * base-layout.component.ts * base-layout.component.css * nav.component.ts * nav.component.html * nav.component.css * footer.component.ts * footer.component.html * footer.component.css * header.component.ts * header.component.html * header.component.css * added routes to the path in app-routing.module.ts * added routes to home page and nav bar and footer * mobile friendly * route: http://localhost:3000/#/ |
| R - 3 | About Us Page  Angular:   * about.component.css * about.component.html * about.component.ts * update routes for nav bar * update routes for app-routing paths * mobile friendly * route: <http://localhost:3000/#/about> |
| R - 4 | Contact Us Page  Angular:   * contact.component.css * contact.component.html * contact.component.ts * iframe for Bellevue university location * mobile friendly * update our routes for nav/landing * update our paths for routing in the app-routing file * route: <http://localhost:3000/#/contact> |
| R - 5 | Sign In Page  Node.js API:   * Employees: * Function: findEmployeeById * Route: /api/employees/:empId   + Example: localhost:3000/api/employees/1007 * Swagger: localhost:3000/api-docs     SoapUI Unit Tests:   * findEmployeeById   Angular:   * signin.component.html * signin.component.css * signin.component.ts * security.service.ts * mobile friendly * Route: localhost:4200/security/signin |
| R – 5.1 | Sign in page   * signin.component.html * signin.component.css * signin.component.ts * security.service.ts * auth.guard.ts * security-routing.module.ts * security.component.ts * has error messages * mobile friendly |
| R - 6 | Tasks Page  Angular:   * tasks.component.css * tasks.component.ts * tasks.component.html * task-management-routing.module.ts * task-mangement.component.ts * task-mangement.module.ts * task.service.ts * employee.interface.ts * item.interface.ts * Route: http://localhost:3000/#/task-management/my-tasks |
| R – 6.1 | Tasks Page  Edited files   * tasks.component.css * tasks.component.ts * tasks.component.html * task-management-routing.module.ts * task-mangement.component.ts * task-mangement.module.ts * task.service.ts * has error messages * mobile friendly   Node.js API:   * Employees: * Function: createTask * Route: /api/employees/:empId/tasks   + Example: /api/employees/1009/tasks   **{**  **"task": {**  **"text": "string",**  **"category": {**  **"categoryName": "string",**  **"backgroundColor": "string"**  **}**  **}**  **}**   * Swagger: localhost:3000/api-docs     Node.js API:   * Employees: * Function: findAllTasks * Route: /api/employees/:empId/tasks   + Example: /api/employees/1009 /tasks   **{**  **"\_id": "64dc00db700fe8a0d1600220",**  **"empId": 1009,**  **"todo": [**  **{**  **"\_id": "64ee9826aff0efa03bfbd942",**  **"text": "testing 2",**  **"category": {**  **"categoryName": "string",**  **"backgroundColor": "string"**  **}**  **}**  **],**  **"done": [**  **{**  **"\_id": "64ee9e07f4c383466c441d23",**  **"text": "testing 45",**  **"category": {**  **"categoryName": "string",**  **"backgroundColor": "string"**  **}**  **}**  **]**  **}**   * Swagger: localhost:3000/api-docs |
| R – 6.2 | Tasks Page  Angular:  Edited files   * tasks.component.css * tasks.component.ts * tasks.component.html * task-management-routing.module.ts * task-mangement.component.ts * task-mangement.module.ts * task.service.ts * tasks.component.css * tasks.component.ts * tasks.component.html * has confirmation and error messages   Node.JS api:   * Function: updateTask * Route: /api/employees/:empId/tasks   + Example:   + /api/employees/1009/tasks * Swagger: localhost:3000/api-docs |
| R – 6.3 | Tasks Page  Edited files   * tasks.component.css * tasks.component.ts * tasks.component.html * asks.component.css * tasks.component.ts * tasks.component.html * task-management-routing.module.ts * task-mangement.component.ts * task-mangement.module.ts * task.service.ts * drag and drop service   Node.js API:   * Employees: * Function: deleteTask * Route: /api/employees/:empId/tasks/:taskId   + Example:   + /api/employees/1009/tasks/64ee9826aff0efa03bfbd942 * Swagger: localhost:3000/api-docs |
| R - 7 | 404 Page   * not-found.module.ts * not-found.module.html * not-found.module.css * added routing to the app-routing.module.ts * path: ‘\*\*’ * route: <http://localhost:3000/#/not-found> |

## 3.3. Feedback/Comments

Make sure the house theme is consistent on all application pages. Use swagger to document APIs and test APIs with SoapUI. Provide function test on various aspects of the application throughout the project.

# SECTION 4: QA TESTING

## 4.1. QA Test Plan

*Sprint 1*

*Test sign in page functionality*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Expected** | **Actual Results** | **(Pass/Fail)** | **Notes** |
| 1 | Navigate to localhost:4200/security/signin | Employee sign in page A screenshot of a login form  Description automatically generated | Employee sign in page | pass | Did as expected |
| 2 | Test if sign in accepts letters | Should give an error  A screenshot of a sign in form  Description automatically generated | Gave an error | Pass | Asdasdasd will give an error |
| 3 | Test random numbers in signin | A screenshot of a login form  Description automatically generated | Gave an error | Pass | 11111 will not be accepted |
| 4 | Test login function with acceptable values 1007-1012 | It will accept the created empId values from the DB | Signed in | Pass | Tested values from the DB and all passed |
| 5 | Check if full name appears | A screenshot of a computer  Description automatically generated | The full name shows | Pass | Tested to see we are able to see full name on which id is logged in |
| 6 | Tested signout | Will log you out of the account and bring you back to the landing page | Logs out of account and directs to landing page | Pass | Checked to see if the signout works as intended |
| 7 | Check to see if return on login page returns user to landing page | Should direct user to landing page | It directs the user to the landing page | Pass | Return works as intended |

*Test task page navigation*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Expected** | **Actual Results** | **(Pass/Fail)** | **Notes** |
| 1 | Navigate to http://localhost:3000/#/ | A collage of images of books and a book  Description automatically generated | Routed to the landing page | pass |  |
| 2 | next navigate to signin in the nav bar | A screenshot of a login form  Description automatically generated | Directs us to the sign in page | pass | if this didn’t work double check the html file route and the app-routing is set up |
| 4 | Sign in with empId 1007 | A screenshot of a computer  Description automatically generated | Routed to the tasks page page | pass | Don’t worry if this has tasks in it or not |
| 5 | click home on the nav bar | A collage of images of books and a book  Description automatically generated | Routed to the landing page | pass | if this didn’t work double check the html file route and the app-routing is set up |
| 6 | On the landing page click the img tasks | A screenshot of a computer  Description automatically generated | Routed to the tasks page page | pass | Don’t worry if this has tasks in it or not |
| 7 | At the top right of nav, select the icon and click sign out | A collage of images of books and a book  Description automatically generated | Routed to the landing page | pass |  |
| 8 | Now click the tasks img | A screenshot of a login form  Description automatically generated | It should redirect us to the signin page | pass |  |

*Mobile friendly test*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Expected** | **Actual Results** | **(Pass/Fail)** | **Notes** |
| 1 | Navigate to http://localhost:3000/#/ | A collage of images of books and a book  Description automatically generated | Routed to the landing page | pass |  |
| 2 | Resize your browers until it matches a mobile phone | A feather on a book  Description automatically generated | it should have a single column of text and images and the hamburger menu replace the traditional nav bar | Pass | If this didn’t work, start with the @media or mess with col and rows, feel free to scroll down and verify the entire page |
| 4 | Open the hamburger menu | A close-up of a feather  Description automatically generated | We should see our nav bar in one column | pass | If this didn’t work, start with the @media or mess with col and rows |
| 5 | Select sign in | A screenshot of a login form  Description automatically generated | An easy one to make mobile friendly should resemble our normal size | pass |  |
| 6 | Sigin in with empId 1007 | A screenshot of a computer  Description automatically generated | We see our form and todo and done in one column | pass | If this didn’t work, start with the @media or mess with col and rows, feel free to scroll down and verify the entire page |

*Sprint 2*

*Contact page*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Expected** | **Actual Results** | **(Pass/Fail)** | **Notes** |
| 1 | Navigate to http://localhost:3000/#/ | A collage of images of books and a book  Description automatically generated | Routed to the landing page | pass |  |
| 2 | next navigate to contact in the nav bar | A screenshot of a map  Description automatically generated | Directs us to the contact us page | pass | if this didn’t work double check the html file route and the app-routing is set up |
| 4 | Click the go home button | A collage of images of books and a book  Description automatically generated | Routed to the landing page | pass |  |
| 5 | Click on the contact us img | A screenshot of a map  Description automatically generated | Directs us to the contact us page | pass | if this didn’t work double check the html file route and the app-routing is set up |

*Create tasks*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Expected** | **Actual Results** | **(Pass/Fail)** | **Notes** |
| 1 | Click task page | Directs you to login | Directs to login | pass | Works as intended |
| 2 | Login with 1010 | Should log in and direct to tasks page | Logged in on the tasks page | pass | Works as intended |
| 3 | Click home on nav bar | It will direct you home | Directs to home page | pass | Works as intended |
| 4 | Click my tasks the account info tab top right | Directs to tasks page | Directs to tasks page | pass | Works as intended |
| 5 | Test the tasks by typing in testing, selecting Testing, click add | You should see a task populated in the todo column, with the color green | A screenshot of a phone  Description automatically generated | pass | Works as intended |
| 6 | Test the tasks by typing in meetings, selecting Meetings, click add | You should see a task populated in the todo column, with the color red | A screenshot of a phone  Description automatically generated | pass | Works as intended |
| 7 | Test the tasks by typing in projects, selecting Projects, click add | You should see a task populated in the todo column, with the color purple | A screenshot of a phone  Description automatically generated | pass | Works as intended |
| 8 | Test the tasks by typing in other, selecting Other, click add | You should see a task populated in the todo column, with no color | A screenshot of a phone  Description automatically generated | pass | Works as intended |
| 9 | Hover over one of the colors | You should see what category was selected | A screenshot of a phone  Description automatically generated | pass | Works as intended |
| 10 | Try to hit submit with no text or category selected | It will not let you hit submit, btn is faded and a message will appear please enter a task | A blue and pink striped background  Description automatically generated | pass | Works as intended |
| 11 | Test the min character length of three | A message will appear, and the btn will remain faded | A blue and white striped background  Description automatically generated | pass | Works as intended |
| 12 | Test the max character length of 50 | A message will appear and the btn will fade | A screenshot of a computer  Description automatically generated | pass | Works as intended |
| 13 | Reload page | You should see the tasks you created in the todo column | A screenshot of a phone  Description automatically generated | pass | Works as intended |
| 14 | add text 4 letters long, and do not select a category | Btn will unfade, however It will not submit and you will get an error message that says to choose a category | A blue and pink striped background  Description automatically generated | pass | Works as intended |

*Mobile friendly*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Expected** | **Actual Results** | **(Pass/Fail)** | **Notes** |
| 1 | Navigate to http://localhost:3000/#/ | A collage of images of books and a book  Description automatically generated | Routed to the landing page | pass |  |
| 2 | Resize your browers until it matches a mobile phone | A feather on a book  Description automatically generated | it should have a single column of text and images and the hamburger menu replace the traditional nav bar | Pass | If this didn’t work, start with the @media or mess with col and rows |
| 4 | Open hamburger menu and select contact | A screenshot of a phone  Description automatically generated | A signle column and a button for calling the Bellevue number, and the iframe | pass | you made need to press the hamburger menu again once page has loaded to get out of menu |

*Sprint 3 functional tests*

*Drag and drop*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Expected** | **Actual Results** | **(Pass/Fail)** | **Notes** |
| 1 | Navigate to http://localhost:3000/#/ | A collage of images of books and a book  Description automatically generated | Routed to the landing page | pass |  |
| 2 | Navigate to tasks page by click on the img | A screenshot of a login form  Description automatically generated | Directs you to the sign in page | pass |  |
| 4 | Type in 1007 | A screenshot of a computer  Description automatically generated | We should now see the tasks page | pass |  |
| 5 | Create a task by adding atleast 3 characters and selecting a single category below the form and hit add | A screenshot of a computer  Description automatically generated | Created a new task | pass |  |
| 6 | Click and hold the task card and drag it to the done column | A computer screen shot of a computer screen  Description automatically generated | The task is now in the done column | pass |  |
| 7 | Click and hold the task card and drag it to the todo column | A screenshot of a computer  Description automatically generated | The task is now in the to do column | pass |  |

*About us*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Expected** | **Actual Results** | **(Pass/Fail)** | **Notes** |
| 1 | Navigate to http://localhost:3000/#/ | A collage of images of books and a book  Description automatically generated | Routed to the landing page | pass |  |
| 2 | Navigate to the about us page from the landing page | A screenshot of a computer  Description automatically generated | It directs us to the about us page | pass |  |
| 4 | Navigate to the home page with the nav bar | A collage of images of books and a book  Description automatically generated | Routed to the landing page | pass |  |
| 5 | Now select the about us page from the nav bar | A screenshot of a computer  Description automatically generated | It directs us to the about us page | pass |  |

*404 error page*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Expected** | **Actual Results** | **(Pass/Fail)** | **Notes** |
| 1 | Navigate to http://localhost:3000/#/ | A collage of images of books and a book  Description automatically generated | Routed to the landing page | pass |  |
| 2 | Next in the URL, make a typo at the end of the URL asdasd  http://localhost:3000/#/asdasd | A screenshot of a computer  Description automatically generated | Directs us to the 404 error page | pass | If this didn’t work, check our app-routing.module.ts and verify our path set to ‘\*\*’ for not-found |
| 4 | Click the go home button | A collage of images of books and a book  Description automatically generated | Routed to the landing page | pass |  |

A test case template with black text

Description automatically generated

## 4.2. Unit Tests

findEmployeeById test

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Expected** | **Actual Results** | **(Pass/Fail)** | **Notes** |
| 1 | Tested 1007 | It should post the results from the DB | A screenshot of a computer  Description automatically generated | Pass |  |
| 2 | Tested 1008 | It should post the results from the DB | A screenshot of a computer  Description automatically generated | Pass |  |
| 3 | Tested 1009 | It should post the results from the DB | A screenshot of a computer  Description automatically generated | Pass |  |
| 4 | Tested 1010 | It should post the results from the DB | A screenshot of a computer  Description automatically generated | Pass |  |
| 5 | Tested 1011 | It should post the results from the DB | A screenshot of a computer  Description automatically generated | Pass |  |
| 6 | Tested 1012 | It should post the results from the DB | A screenshot of a computer  Description automatically generated | Pass |  |

createTasks

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Expected** | **Actual Results** | **(Pass/Fail)** | **Notes** |
| 1 | Create a post for 1009  {  "task": {  "text": "text test",  "category": {  "categoryName": "test",  "backgroundColor": "red"  }  }  } | It should how the mongo Id for created task | A screenshot of a computer  Description automatically generated | Pass | It worked as intended |
| 2 | Attempt to post something other than the original schema | 400 error | A screenshot of a computer  Description automatically generated | Pass | It worked as intended |
| 3 | Attempted to add more properties | 400 error | A screenshot of a computer  Description automatically generated | Pass | It worked as intended |

findAllTasks - updated

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Expected** | **Actual Results** | **(Pass/Fail)** | **Notes** |
| 1 | Create a GET request for 1009 | It should show all the tasks for that empId | A screenshot of a computer code  Description automatically generated | Pass | It worked as intended |
| 2 | Try an empId that doesn’t exist example 1111 | 404 error | A screenshot of a computer  Description automatically generated | Pass | It worked as intended |

updateTasks

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Expected** | **Actual Results** | **(Pass/Fail)** | **Notes** |
| 1 | Under createTasks api, double click request 1 | A screenshot of a computer  Description automatically generated | It will open a window for task creation with the method of post | Pass | if you don’t see this window, check that you are selecting request 1, if you open the parents it will show a different window |
| 2 | In resource path: replace where it would say empId or replace the current empId with 1012 and press the green arrow at the top right of the window | A screenshot of a computer  Description automatically generated | We see an Id string populate | Pass | If you get an error check the path name, or the employee.json file path and make sure it matches, also make you are using a valid empId |
| 3 | In resource path: replace where it would say empId or replace the current empId with 1012 and press the green arrow at the top right of the window | A screenshot of a computer  Description automatically generated | We see an Id string populate (yes we really are doing this multiple times) | Pass | If you get an error check the path name, or the employee.json file path and make sure it matches, also make you are using a valid empId |
| 5 | under findAllTasks, double click request 1 | A screenshot of a computer  Description automatically generated | We should see a window with the method get | Pass | if you don’t see this window, check that you are selecting request 1, if you open the parents it will show a different window |
| 6 | Replace empId or the actual empId with 1012 under rescource | A screenshot of a computer  Description automatically generated | We should see our three tasks we created | pass | If this doesn’t show up, verify the empId is correct or verify the resource is correct or the path of the employee.json file |
| 7 | Under updateTasks, select request 1 | A screenshot of a computer  Description automatically generated | We will see a window with the method put with filler text | pass | if you don’t see this window, check that you are selecting request 1, if you open the parents it will show a different window |
| 8 | Go back to your findAlltasks window, and highlight then copy an object with CTRL + C | A screenshot of a computer  Description automatically generated | It should be in our clipboard to be posted in the next step | pass |  |
| 9 | With your updateTasks api open, paste into the todo object | A screenshot of a computer  Description automatically generated | There should be our task populated in the todo object | Pass |  |
| 10 | Go back to your findAlltasks window, and highlight then copy the second object with CTRL + C | A screenshot of a computer  Description automatically generated | It should be in our clipboard to be posted in the next step | pass |  |
| 11 | With your updateTasks api open, paste into the done object | A screenshot of a computer  Description automatically generated | There should be our task populated in the done object | pass |  |
| 14 | In our updateTasks api, verify the method is post and replace the :empId with a valid empId of 1012, then press the green arrow at the top right | A screenshot of a computer  Description automatically generated | In the raw file you will see a 204 message | pass | NOTE: change the 204 message from No Content to update successful |
| 15 | We can then verify if we go to findAllTasks again and type in 1012 in empID | A screenshot of a computer  Description automatically generated | And you will now see one of the objects in the done column | pass | Second way of verifying |

deteleTasks

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Expected** | **Actual Results** | **(Pass/Fail)** | **Notes** |
| 1 | Select createTasks and double click request 1 | A screenshot of a computer  Description automatically generated | We see a window with the POST method | pass | if no post method, verify it’s the correct api then select it as post |
| 2 | Replace the empId with 1011 and press the green arrow at top right | A screenshot of a computer  Description automatically generated | It will give you an id | Pass | if you don’t see the json file, double check your in the correct api, if its still blank you may need to add it |
| 4 | Highlight and copy with the CTL + C command between the “ “ | A screenshot of a computer  Description automatically generated | It will be added to the clipboard | pass |  |
| 5 | Select deleteTasks and double click request 1 | A screenshot of a computer  Description automatically generated | We will see a method of delete and a resource that will accept a long id at the end and an empId | pass |  |
| 6 | Paste the id at the end of api path, and replace the empId with 1011 and hit the green arrow at the top right of the window | A screenshot of a computer  Description automatically generated | Under the raw tab it should show 204 no content confirming that there is no longer that Id | Pass |  |

A screenshot of a computer

Description automatically generated

findEmployeeById test Swagger

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Expected** | **Actual Results** | **(Pass/Fail)** | **Notes** |
| 1 | Navigate to http://localhost:3000/api-docs/ | A screenshot of a computer  Description automatically generated | Page loaded correctly with API findEmployeeById | Pass | It worked as intended |
| 2 | Click GET click Try it out  Add 1009 | A screenshot of a computer  Description automatically generated | findEmployeeById pulls up Ally’s info | Pass | It worked as intended |
| 3 | Click GET click Try it out  Add 1111 | A screenshot of a computer  Description automatically generated | 404 | Pass | It worked as intended |

Create task and get task by empid - corrected

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Expected** | **Actual Results** | **(Pass/Fail)** | **Notes** |
| 1 | Navigate to http://localhost:3000/api-docs/ | A screenshot of a computer  Description automatically generated | Page loaded correctly | Pass | It worked as intended |
| 2 | Click POST, click try it out and type 1010, can even fille the “string” values with some message | A screenshot of a computer  Description automatically generated | createTask created a new task for Harper’s profile | Pass | It worked as intended |
| 3 | Click GET empid by tasks click Try it out  Add 1010 | A screenshot of a computer  Description automatically generated | It shows all the tasks from the empId | Pass | It worked as intended |

**updateTasks**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | Navigate to http://localhost:3000/api-docs/ |  | The entire swagger page of all our APIs | Pass | You see exactly whats expected |
| 2 | Navigate to POST |  | Tab opens up for the POST | pass | The post form should show exactly whats in the photo |
| 3 | Click try it out | It will allow you to type in the form | It lets you type in the form | pass | You will be unable to test the api with out first clicking the try it out button |
| 4 | Type in 1007 for empId, and edit the task information {  "task": {  "text": "test ",  "category": {  "categoryName": "test",  "backgroundColor": "test"  }  }  }, Create at least two of these, test 1 and test 2 and click post | A screenshot of a computer  Description automatically generated | If successful you should see a message and a 201 message | Pass | If it fails that means we created either the schema or the entire swagger doc in the employee.json file wrong |
| 5 | Navigate to GET empId/tasks |  | Tab opens up for GET task by empID | pass | If it fails that means we created either the schema or the entire swagger doc in the employee.json file wrong |
| 6 | Click try it out | It will allow you to type in the form | It lets you type in the form | pass | You will be unable to test the api with out first clicking the try it out button |
| 7 | Type in 1007 for empID | This will populate a list of created tasks |  | pass | If it fails that means we created either the schema or the entire swagger doc in the employee.json file wrong |
| 8 | Copy a task from the TODO colomn | Highlight and CTL + C the |  | pass |  |
| 9 | Navigate to the PUT api |  | You should see the entire epi once you open the tab | pass |  |
| 10 | Click try it out, and type in 1007 for empID, and paste the task into the todo section |  | You will see the todo has new text to match the information from the empId by tasks | pass |  |
| 11 | Type in 1007 for empID | This will populate a list of created tasks |  | pass | If it fails that means we created either the schema or the entire swagger doc in the employee.json file wrong |
| 12 | Copy a task from the TODO column (a different one than previously) | Highlight and CTL + C the |  | pass |  |
| 13 | Navigate to the PUT api |  | You should see the entire epi once you open the tab | pass |  |
| 14 | Click try it out, and type in 1007 for empID, and paste the task into the todo section |  | You will see the done has new text to match the information from the empId by tasks | pass |  |
| 15 | Press execute |  | You should see a code 204 tasks updated successfully and see the api change from todo to done for one task | pass | If it fails that means we created either the schema or the entire swagger doc in the employee.json file wrong |

**deleteTask**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Expected** | **Actual Results** | **(Pass/Fail)** | **Notes** |
| 1 | Navigate to http://localhost:3000/api-docs/ | A screenshot of a computer  Description automatically generated | The entire swagger page of all our APIs | Pass | You see exactly whats expected |
| 2 | navigate to post | A screenshot of a computer  Description automatically generated | You should see the api for post | pass | If you don’t see this you created the employee.json file wrong for this api |
| 3 | Click try it out and create a task like this, and enter the empID 1007 {  "task": {  "text": "test 3",  "category": {  "categoryName": "test",  "backgroundColor": "test"  }  }  }  click execute | A screenshot of a computer  Description automatically generated  A screenshot of a computer  Description automatically generated | You should populate the information needed to create a task and once submitted see a 201 task created | pass | If you get an error message the api was created wrong or setup wrong inside the employee.json file |
| 4 | Highlight and copt the ID inside the pair of quotes | A black and white stripe  Description automatically generated | Highlighted text and CTL + C | Pass | This is a step to prep for the next step |
| 5 | Navigate to the DELETE API | A screenshot of a computer  Description automatically generated | Saw what was expected | Pass | If you don’t see this you created the employee.json file wrong for this api |
| 6 | Click try it out and type in empID 1007, and paste your id within the taskID and press execute | A screenshot of a computer  Description automatically generated | We see the code 204, which checks to see if it was deleted | Pass | If you get an error message the api was created wrong or setup wrong inside the employee.json file |

# SECTION 4: REFERENCES

Miroshnichenko, T. (2020, October 8). *Man using a laptop · Free stock photo*. Pexels. https://www.pexels.com/photo/man-using-a-laptop-5198239/

Pexels. (2020, October 2). *Man using a laptop · Free stock photo*. https://www.pexels.com/photo/man-using-a-laptop-5483063/

Piacquadio, A. (2020, May 2). *Confident businesswoman sharing information from documents in workplace · Free Stock Photo*. Pexels. https://www.pexels.com/photo/confident-businesswoman-sharing-information-from-documents-in-workplace-3757946/