Table of Contents

Software Engineer Foundation Assignment #1

Start on week 2

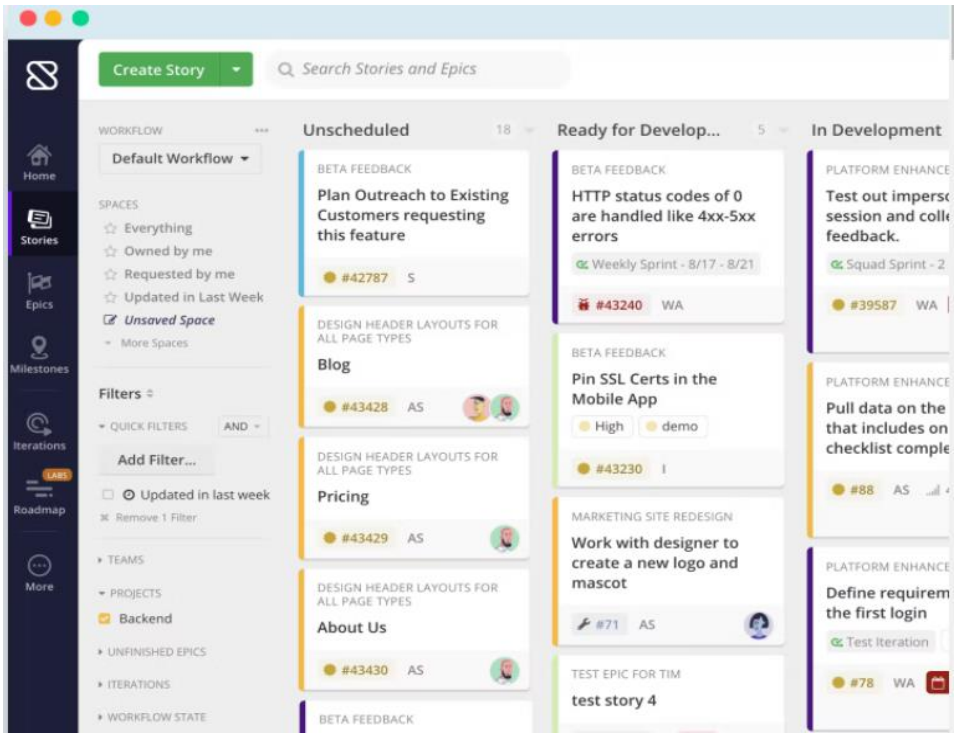
The objective of this assignment is to expose the SW engineer to the fundamental of developing an application that covers the following:

1. Identity and access control of an application
2. State transition with pre & post actions to be triggered for a workflow which is usually implemented in projects. Eg. Case management, eSOP, etc. The pre & post actions contain the process/business rules, constraints for the respective business workflows
3. REST API for system integration purposes
4. Containerize the Nodejs application and API

In this assignment we shall build a Task Management System that emulate the Kanban approach. A kanban board is an agile project management tool designed to help visualize work, limit work-in-progress, and maximize efficiency (or flow). It can help both Agile and DevOps teams establish order in their daily work. Kanban boards use cards, columns, and continuous improvement to help technology and service teams commit to the right amount of work, and get it done! Please refer to the following URL for more info.

> https://www.atlassian.com/agile/kanban/boards#:~:text=A%20kanban%20board%20is%20an,maximize%20efficiency%20(or%20flow).&text=Kanban%20boards%20use%20cards%2C%20columns,work%2C%20and%20get%20it%20done

The following diagram depicts an example of a digital Kanban in a commercial software.



Assignment 1 of 4: To implement the user management features

There are 3 sections to this assignment which include the following:

1. Set up node.js and respective modules on your windows PC
2. Copy the source code from provided URL and make the necessary changes to perform user authentication using MySQL database
3. Extend the source code from part 2 to implement a feature to update the user info as part of a user management function.

1.1 Environment Set up

Set up your development environment on your windows or Linux PC.

Basic environment setup

Install Node.js, MySQL, and Node.js respective modules.

- Visual Source Code (this is your IDE)

  https://code.visualstudio.com/download

- MySQL Server (this is your Database)

  Install MySQL Server on your windows PC with the workbench option. You can refer to the following url for the installation guide.
  https://www.udemy.com/tutorial/sql-tutorial-learn-sql-with-mysql-database-beginner2expert/download-install-mysql-8011-on-windows-10-operating-system/

- Node.js (this is your App server)

  Install Node.js per instruction @ https://nodejs.org/en/

  For the following set up, launch powershell in windows (or terminal in Linux) and execute the following instructions

- Express - Install with command: `npm install express`.
- Express Sessions - Install with command: `npm install express-session`.
- MySQL for Node.js - Install with command: `npm install mysql`.

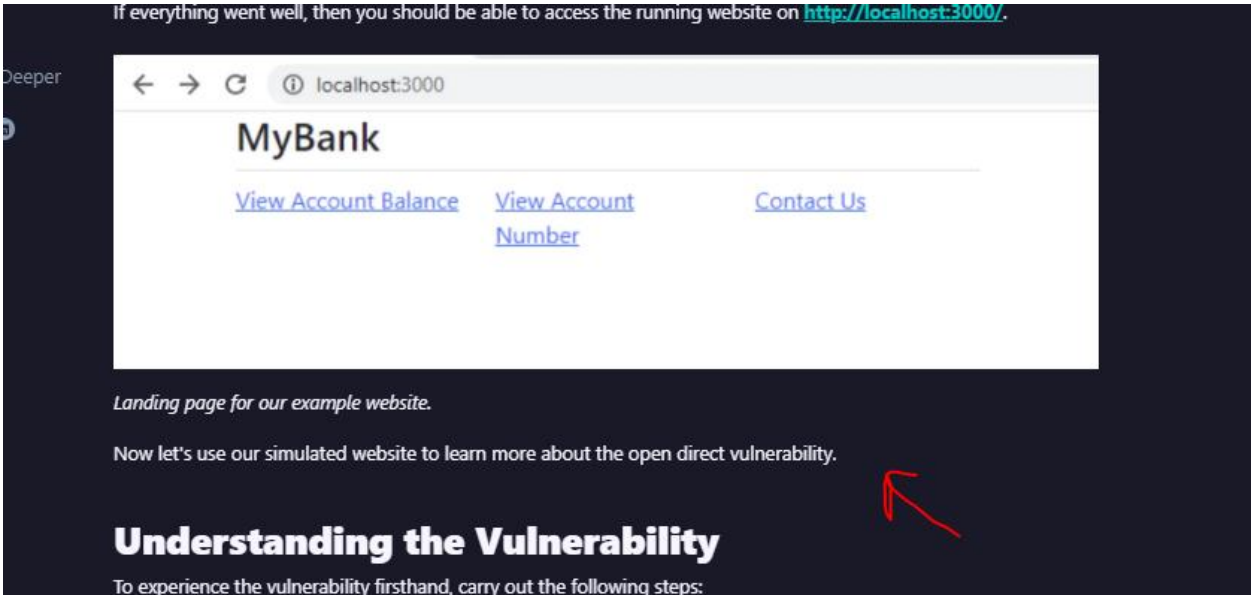
Create the base source code for the assignment

Create the default source code using the following URL.

Note: Some of the installation could already be done in the setup as above.

https://www.stackhawk.com/blog/nodejs-open-redirect-guide-examples-and-prevention/#creating-the-nodejs-file

You just need to follow the instructions to set up and create the files per instructed till the step indicated by the red arrow in the diagram below.



1.2 Establish the Database connection

Using the source code created in part 1, you are required to make changes to the source code in Node.js file. The current authentication is using the following code.

```
app.post('/login', (req, res) => {
  const {username, password} = req.body;
  if (username === 'bob' && password === '1234') {
    req.session.isLoggedIn = true;
    res.redirect(req.query.redirect_url ? req.query.redirect_url : '/');
  } else {
    res.render('login', {error: 'Username or password is incorrect'});
  }
});
```

You are required to change this hard coded username and password with the username and password maintain in MySQL database. (https://codeshack.io/basic-login-system-nodejs-express-mysql/)

Please run the following database script using MySQL workbench to i) create the database (nodelogin), 2) create the table accounts, and 3) insert the username=test, password=test

```
CREATE DATABASE IF NOT EXISTS `nodelogin` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
USE `nodelogin`;

CREATE TABLE IF NOT EXISTS `accounts` (
```

```
  `id` int(11) NOT NULL,
  `username` varchar(50) NOT NULL,
  `password` varchar(255) NOT NULL,
  `email` varchar(100) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;


INSERT INTO `accounts` (`id`, `username`, `password`, `email`) VALUES (1, 'test',
'test', 'test@test.com');


ALTER TABLE `accounts` ADD PRIMARY KEY (`id`);
ALTER TABLE `accounts` MODIFY `id` int(11) NOT NULL
AUTO_INCREMENT,AUTO_INCREMENT=2;
```

The above instructions are meant to help you get started and should you be required to focus on ReactJS, you would need to look at the various implementation of the front-end application design that will impact the application architecture. (ie. Monolith vs layered)

1.3 Create the user management functionalities

In this module, you will learn and apply the CRUD (create, read, update, delete) concept to the database related functions & features to be developed. You will be required to develop the front-end and back-end aspect of the user management use case described in the following sections.

1.3.1 Basic user management

Using the source code updated in section 1.2, you are required to implement the following as part of the Task Management System to be built:

1. Create the login screen
2. Create a main menu screen with the “User Management” Option
3. Create the following functions under the User Management
    a. Create new User
    b. Change password for current login user
    c. Update email information
4. Repeat step 1 to show the email is updated.

1.3.2 Basic user group management

In this section, you are required to create the User Group functionality. For this section, you are only required to develop the user group to be associated to user account. (group within group is not necessary for a start but you might need this in future).

You would need to create a function that returns a value to indicate if a user is in a group. Eg. Checkgroup(userid, groupname)

1.3.2 Privilege user management & Security

In this section, you would have to develop the security aspect of the User Management use case as follows:

1. Create admin privilege account(s) to allow:
    a. Create user
    b. Disable user
        i. Note: you would need to create a field in the database to indicate the user account is disabled or active. Deletion of user account is not allowed as this information could be used in the Task Management activities and audit trail is needed even if the user is no longer allowed to access the application.
    c. Reset other user account password and related information.
        i. Note: you cannot allow user id to be changed as per para (1.b.i)

    Note: You can design an "admin" account which we assume is part of the Task Management application set up by default. You would need to

2. Ensure the password is not stored in clear text in the database which is a violation to security requirement. You can explore the use of password hashing approach which is a single direction which does not support the recreation of the password. IE, you will compare the hashed password for authentication. You can consider the use of the bcrypt or argon2 module.
    https://www.npmjs.com/package/bcrypt
    https://www.npmjs.com/package/argon2
    https://www.reddit.com/r/node/comments/qy3608/password_hashing_with_bcrypt_vs_bcryptjs_vs/

3. Ensure the password comprises of the following characteristics
    a. Minimum 8 characters and maximum 10 characters
    b. Comprise of alphabets , numbers, and special character

Assignment 2 of 4: To implement the Task Management features

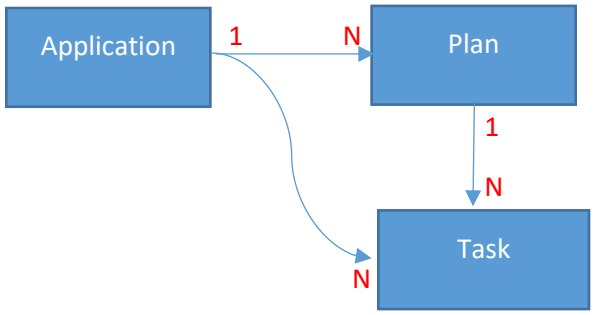
Start on week 4

The objectives of this assignment are to:

1. Understand the workflow needed to implement the lifecycle of a "task" which is akin to any case management related system.
2. Understand the custom fields need to be captured as data to support the "task".
3. Understand the "screen" which the custom fields shall be associated to as the interface to the system.
4. Understand how to implement the Task Management features related to specific state of the workflow in terms of access management, pre-validation, and post-action needed for the state transition to occur. (across multiple SQL tables for CRUD transactions)

2.1 Defining the Task Management System

The Task Management System (hereafter known as TMS) to be build shall support the Planning, Tracking, and Approving for the task activities related to a specific project. The project shall be described as follows and shall describe the database data model (covered in 2.1.2):

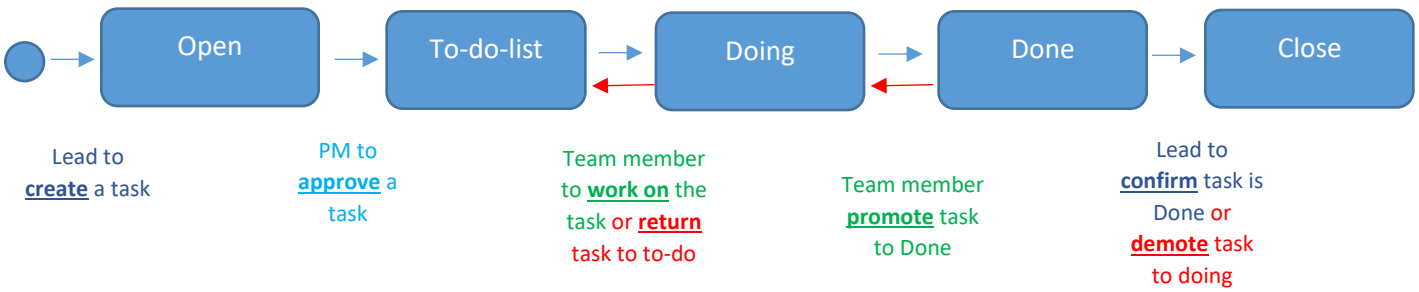


2.1.1 Defining the Workflow

To define the workflow of TMS, we shall identify the persona involved in the user stories. The user stories includes the following:

a. As a project lead, I would need to define and store my applications information for my project to be used to trace to all the Plans and Tasks to work on.
b. As a project manager, I would need to plan all the scheduled MVP(s) with the start and end date as per my project plan/schedule.
c. As a project lead, I would need to define all the tasks that is required in X sprints needed to deliver the MVP.
d. As a team member, I would need to work on the tasks and seek approval to close the task from the team lead when done.

You are required to implement the workflow for the “Task” only. State indicated as the box and arrow is the action taken (bold & underlined text) to trigger a state transition.



You are required to:

1. Implement which user-group can perform the state transition as per workflow (pre-validation). The user-group shall be configurable for each "Application" team within the project. (see the Application Table below. Eg. User-group permitted to create the task is defined in App_permit_create,)
2. Implement an email notification to the lead when the team member has promoted the task to "Done" state (post-action)
3. Implement a task-id using the format of [application_acronym]_[running number]. The task-id is assigned whenever a task is created (open state). IE. The task-id shall be system generated and always in read-only mode.
4. Implement an audit trail (read-only) of the notes provided throughout the life-cycle of the task. The notes shall be stamped with a. logon userid, b. current state, c. date & timestamp.

2.1.2 Defining the Data Model and Custom Fields

The TMS data model shall be represented in your MySQL as follows:

Note: The colour code in the field represents the relationship. *App_Acronym*, *Plan_MVP_name*, and *Task_name* shall only be provided during its creation and shall remain as read-only thereafter since this would be the table key.

Table: Application	Table: Plan	Table: Task
Field	Field	Field
App_Acronym	Plan_MVP_name	Task_name
App_Description	Plan_startDate	Task_description
App_Rnumber	Plan_endDate	Task_notes
App_startDate	Plan_app_Acronym	**Task_id** <App_Acronym>_<App_Rnumber>
App_endDate		Task_plan
App_permit_Open		Task_app_Acronym
App_permit_toDoList		Task_state
App_permit_Doing		Task_creator
App_permit_Done		Task_owner
		Task_createDate

2.1.3 Defining the Screen

In this assignment, you are given the autonomy to decide how you would design the UI of TMS. However, do keep in mind the UX aspect to ensure the ease of use and efficiency factor.

Assignment 3 of 4: To implement 2 REST API for TMS

Start on week 6

The objectives of this assignment are to:

1. Understand how REST API operates from the security and assessment aspect

2. Understand how REST API can be designed to be scalable

3.1 CRUD transactions

In this assignment you would have to create the following API on the nodejs platform.

a. Create a new task (method name = CreateTask)
b. Retrieve tasks in a particular state (method name = GetTaskbyState)
c. Approve a task from "Doing to Done" state (method name = PromoteTask2Done)

You can use Postman to test your API. (https://www.postman.com/downloads/) Postman is an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs—faster.

As you design and build these API, you would need to consider the scope of what each API does so as to extend these implementations to Microservices design if possible. Since the tenet of microservices focus on scalability with a limited scope of work. However, do make sure that you do not fall into the pitfall of creating a mini-service as a result of using the "common code / shared services" approach which result in wastage of processing and memory requirements when deploying these "services" in the Kubernetes pod.

You shall document your API in the following format

Name	Method	Parameters	Behaviour	Expected Output format

Eg. CreateTask	post	-u username	Mandatory	Json object:
				{
				"task_id":"APPLE_3",
				"code": "200"
				}

		-p password	Mandatory	
		-a application acronym	Mandatory	
		-n name of task	Mandatory	
		-d description of task	Optional	
Example				
1. <show an example of how to call your API using javascript> code: …… Output: ……. 2. <show an example of how to call your API using curl> code: …… Output: …….				
Error Code				
<List the error code returned>				

During your implementation of the API, you would need to consider what are the potential test cases with the consideration of the test data.

Test data

Before performing a test, you need to decide what data you are going to include in your test case. It is not normally possible to perform tests with every single possible piece of data. So, instead the developers will choose from a limited range of data such as:

- **valid data** - sensible, possible data that the program should accept and be able to process
- **extreme data** - valid data that falls at the boundary of any possible ranges
- **invalid (erroneous) data** - data that the program cannot process and should not accept

Tests should find that the program works as expected. Obvious input data should confirm that the software works as expected. Extreme test data will be chosen to test what breaks the system.

Assignment 4 of 4: To containerize the APIs in assignment 3

Start on week 7

The objectives of this assignment are to:

1. Understand how containerize the API using dockerfile and relevant considerations
2. Understand how to reduce the size of your container image using a streamlined OS without effecting the function of the API to be delivered.

4.1 Set up Docker

In this assignment you would have to perform the following:

I. Install Docker (engine)
   a. https://docs.docker.com/desktop/windows/install/

II. containerize your API on node.js
   a. https://nodejs.org/en/docs/guides/nodejs-docker-webapp/
      i. Create a dockerfile
      ii. Identify the builder image from docker hub to use
      iii. Build your image
      iv. Import your image into Docker Desktop
      v. Define your Container/App in Docker Desktop
      vi. Execute the Container to start the APIs
      vii. You are required to build your application container image in less than 200Mb in size.

Note: Consider the size of the application container image and how you will transport this image to another environment (air-gap) with different database connection information. You would also to ensure you are not using the root account when running your application container image for security purpose.

Assignment Optional: Performance and Security

5.1 Database Connection (Performance)

As a developer, you may not have spent a lot of time thinking about database connections. A single database connection is not expensive, but as things scale up, problems can emerge.

Reading materials: https://medium.com/dscjssstu/pooling-connections-in-node-js-mysql-9685d5c03c30

Sample Tutorial: https://codeforgeek.com/node-mysql-connection-pool-example/

5.2 Application Access Control (Security)

As part of some sensitive application implementation, you would most likely encounter the following requirements.

1. Ensure there is a user timeout capability (fixed or customizable)
2. Ensure the application does not allow a user to have more than 1 concurrent login
3. Ensure there is a user login/logout audit trail