

# **Collaborative API Software : AlterAPI**

**Submitted in Partial Fulfilment of the Requirements for the Award of the Degree of**

**MASTER OF COMPUTER APPLICATIONS (M.C.A)**



*Submitted By*

**DEEPAK SHARMA**  
**Enrollment No : 02611404419**

**Guided By**

**Internal Guide**

Mr. Shrikant Patel  
Assistant Professor

**External Guide**

Mr. Prashant Sharma  
Axeno Consulting Pvt. Ltd.



**Department of computer science and application**  
**Bhai Parmanand Institute of Business Studies**  
**Guru Gobind Singh Indraprastha University, Delhi**  
**Session: (2019-2022)**

**GOVERNMENT OF NCT OF DELHI  
BHAI PARMANAND INSTITUTE OF BUSINESS STUDIES**

**Opp. Madhuban Colony: Shakarpur Extn. Delhi-110092**

**Email: bpibs.delhi@nic.in**

**Phone: 22543891, 22017393 Fax: 22016134**



**Certificate by the Head of Department**

This is to certify that this project report entitled “**Collaborative API Software : AlterAPI**” is submitted as the diligent work of **Deepak Sharma (Enrollment No. – 02611404419)**, who is undergoing 6<sup>th</sup> Semester Industrial training at **Axeno Consulting Pvt. Ltd.** He has been given regular updates about the training work throughout the semester and has prepared the training report under the guidance of the assigned guide.

**Date:**

**Dr. Girish Sharma  
(Professor & Head of MCA Department)**

**GOVERNMENT OF NCT OF DELHI**

**BHAI PARMANAND INSTITUTE OF BUSINESS STUDIES**

**Opp. Madhuban Colony: Shakarpur Extn. Delhi-110092 Email: bpibs.delhi@nic.in Phone:  
22543891, 22017393 Fax: 22016134**



### **Certificate by the Internal Guide**

This is to certify that this project report entitled “**Collaborative API Software : AlterAPI**” is submitted as the diligent work of **Deepak Sharma ( Enrollment No. – 02611404419 )**, who is undergoing 6<sup>th</sup> Semester Industrial training at **Axeno Consulting Pvt. Ltd.** He has been given regular updates about the training work throughout the semester and has prepared the training report under my guidance.

**Date:**

**Project Guide**

**Mr. Shrikant Patel**

**(Assistant Professor)**

**GOVERNMENT OF NCT OF DELHI**

**BHAI PARMANAND INSTITUTE OF BUSINESS STUDIES**

**Opp. Madhuban Colony: Shakarpur Extn. Delhi-110092 Email: bpibs.delhi@nic.in Phone:  
22543891, 22017393 Fax: 22016134**



## **CANDIDATE'S DECLARATION**

I hereby declare that the work which is being presented in this project work entitled **“Collaborative API Software : AlterAPI”** in partial fulfillment of the requirements for the award of the degree of Master in Computer Applications at Bhai Parmanand Institute of Business Studies, is an authentic record of my own work carried out during the period April to June 2022 at **Axeno Consulting Pvt. Ltd.** under the supervision and guidance of Mr. Prashant Sharma (Training Head).

I assure you that I have not submitted the matter embodied in this project work anywhere for the fulfillment of any degree or diploma.

Date:

New Delhi

Deepak Sharma ( 02611404419 )

Signature

The Principal  
BPIBS,  
Shakarpur, Delhi-92

### **UNDERTAKING - MCA DISSERTATION**

I, **Deepak Sharma** Roll No. **02611404419** may kindly be sponsored to **Axeno Consulting Pvt. Ltd.**, for Practical Training- Software/ Project Preparation.

I hereby undertake that:

1. I will abide by the rules of the organisation and observe total discipline as per their conditions.
2. I will follow all the rules, regulations and ordinances of the GGS Indraprastha University, Delhi.
3. I will prepare and submit the duly completed Software/Project in the institute along with requisite reports well in time.
4. I will arrange to remain present in the institute for presentation discussion and viva as per schedule of internal/ external examination.

Yours faithfully

Date: MAY 15 2020

Signature (\_\_\_\_\_)

Name: **Deepak Sharma**

Complete Address:

**E-83, Street Number -3, First  
Floor, East Vinod Nagar.**

**New Delhi-110092**

Tele. No.: +91 9540788379

## PROFORMA OF CERTIFICATE FOR PROJECT

This is to certify that is a bonafide record of the project work done satisfactorily at **Axeno Consulting Pvt. Ltd.** by Mr. **Deepak Sharma** Registration No. **02611404419** in practical fulfillment of MCA 6<sup>th</sup> Semester Examination.

This report or a similar report on the topic has not been submitted for any other examination and does not form part of any other course undergone by the candidate.

SIGNATURE ( )

(Candidate)

PLACE: New Delhi

NAME: Deepak Sharma

DATE: 15 May 2022

ROLL NO: 02611404419

SIGNATURE ( )

(Project Guide)

PLACE: New Delhi

NAME: Mr. Shrikant Patel

DATE:

DESIGNATION: Assistant Professor

ADDRESS: Opp. Madhuban Colony, Near Vikas  
Marg, Shakarpur,  
New Delhi, Delhi 110092

Name & Seal of the Institution

# ACKNOWLEDGEMENT

It is my proud privilege to express my profound gratitude to the entire management of Bhai Parmanand Institute of Business Studies & faculty members of the institute for providing me with the opportunity to avail the excellent facilities and infrastructure. The knowledge & values inculcated in me during the academic session 2019-2022 have proved to be of immense help at the starting of my career. **I am very thankful to our Principal Mr. Girish Sharma for providing us an excellent infrastructure & experienced faculty members with immense knowledge at BPIBS.**

I am grateful to my project guide Assistant Prof. **Mr. Shrikat Patel** for their astute guidance, constant encouragement & sincere support for this project work. Without them it was impossible to understand the use cases of **Collaborative API Software : AlterAPI**

I would like to thank our Training & Placement Officer **Mr. Kaushal Mehta** for arranging placement drives . I would like to thank Edunet Foundation, for providing me with an opportunity to pursue my Artificial Intelligence training. I feel pride and privilege in expressing my deep sense of gratitude to all those who have helped me in presenting this assignment.

I express my sincere gratitude to **Axeno Consulting Pvt. Ltd.** and **Mr. Prashant Sharma** for his inspiration, constructive suggestion, mastermind analysis and affectionate guidance in my work, without which this project work completion would have been impossible for me.

In the end, I am really thankful to all the faculty members of BPIBS & my colleagues . Without their support & belief, I couldn't reach the place where I am today. I am feeling overwhelmed today.

# Table of Contents

## Preface

Serial No.	Description	Page no.
1	<b>Introduction : Software Requirements Specification</b>	2
	1.1 Problem Statement	3
	1.2 Proposed Solution	6
	1.3 Project Scope	7
	1.4 Features	8
2	<b>Project Description</b>	9
	2.1 Methodology Used	11
	2.2 Technology Used	14
	2.3 Required Tools	19
	2.4 System Requirements	20
3	<b>System Analysis</b>	21
	3.1 Data Flow Diagram	22
	3.2 Use Case Description and Diagram	26
	3.3 ER Diagram	30
	3.4 Class Diagram	31
	3.5 State Transition Diagram	32
4	<b>System Design</b>	33
	4.1 Database Design	34
	4.2 Project Structure	35
	4.3 Coding	39
	4.4 Form Design	40
	4.5 UI Design	45
5	<b>Project Management</b>	49
	5.1 Project Planning and Scheduling	52
	5.1.1 Gantt Chart	
	5.2 Risk Management	53



6	<b>System Testing and Deployment</b>	56
	6.1 Testing	57
	6.2 Deployment	64
7	<b>Result and Conclusion</b>	65
	7.1 Conclusion	66
	7.2 Limitation	67
	7.3 Future Scope of Project	67
	7.4 Reference	68

# Table of Figures

This table contains a list of all diagrams, Figures and related Images.

Serial No.	Description	Page no.
1	<b>Introduction : Software Requirements Specification</b>	
	1.1 Working of API	4
2	<b>Project Description</b>	
	2.1 Trello Kanban (a)	13
	2.2 Trello Kandab (b)	13
	2.3 Frontend Technologies	16
	2.4 Backend Technologies	17
	2.5 NoSql Database Technology	18
	2.6 Firebase Hosting	18
	2.7 Tools used for developing application	19
3	<b>System Analysis</b>	
	3.1 Level 0 DFD	22
	3.2 Level 1 DFD	23
	3.3 Level 2 DFD - User Authentication	24
	3.4 Level 2 DFD - Workspace	25
	3.5 Use Case Diagram	29
	3.6 ER Diagram	30
	3.7 Class Diagram	31
	3.8 State Transition Diagram	32
4	<b>System Design</b>	
	4.1 Database Design	34
	4.2 Package.json file (Dependencies)	35
	4.3 Project Structure - app folder	36

	4.4 File Structure	37
	4.5 Project Structure	38
	4.6 Login Form design	40
	4.7 Register Form design	41
	4.8 Create Workspace Form design	42
	4.9 Add Request Form (a)	43
	4.10 Add Request Form (b)	43
	4.11 Invite User Form design	44
	4.12 Landing Page (About)	45
	4.13 Landing Page (Feature)	45
	4.14 Workspace List	46
	4.15 Invitation Page	46
	4.16 Request Tab Page (a)	47
	4.17 Request tab Page (b)	47
	4.18 Documentation Page (a)	48
	4.19 Documentation Page (b)	48
5	<b>Project Management</b>	
	5.1 Gantt Chart	52

# PREFACE

This dissertation is an original intellectual product of the author, **Deepak Sharma**. Which provides the detailed description of “**Collaborative API Software: Alter-API**”.

It includes detailed description of technologies which are being used in **Alter-API** the project work on which trainee have worked upon during the training period April 1,2022 to May 25, 2022. Actual tenure of training is April 1, 2022 to July 30, 2022.

The following is a brief description of the points that are covered in each section throughout the report :

**Chapter 1** is an Introduction to the project, which covers problem statements, proposed solutions , project scope and process model.

**Chapter 2** includes the project description in detail, containing methodology used , technology used for building the software and different tools required for the development.

**Chapter 3** is all about the System Analysis which includes analysing the requirements, designing DFDs, ER, Use Case, Class, State Transition diagram. With these diagrams Use case description is also mentioned for building use case diagrams.

**Chapter 4** is of System Design , it covers File / Database Design, Project Structure, Coding and different form designs. It also includes UI Design screenshots of how the application looks like and users interact with it.

**Chapter 5** includes the Project Management by planning and scheduling, it contains a timeline chart of project development. It also includes a Risk management system.

**Chapter 6** is all about the Testing and Deployment of the application, different levels of testing performed in the project along with respective entities, and how deployment activity is carried out.

**Chapter 7** concludes the project, limitation of project, future scope of the project, useful resource links and references.

# **Chapter 1**

## **INTRODUCTION**

1.1 Problem Statement

1.2 Proposed Solution

1.3 Project Scope

1.4 Features

Project titled “**Collaborative API Software : Alter-API**” *provides a collaborative environment for developers to build, test, document and publish APIs with ease.*

## **Software Requirements Specification :**

A software requirements specification (SRS) is a description of a software system to be developed. It is modelled after business requirements specification. The SRS lays out functional and non-functional requirements, and it may include a set of use cases that describe user interactions that the software must provide to the user for perfect interaction.

Now, What are the software requirements for “**Collaborative API Software : Alter-API**” ? In the 21st Century technology is evolving and improved every day to have a seamless user experience. As we focus on developing Applications, new tech stacks and frameworks are introduced every year with a fast and easy development process. Developers now have many options to choose from which technology to use for which project and how it needs to be implemented. So, this project is among those technologies that helps developers to be more productive in building *APIs*.

### **1.1 Problem Statement**

*APIs? What is it? It has been mentioned many times till now. Knowing it helps in understanding What project is all about.*

Application Programming Interface is just a way of communicating between two applications. It enables applications to exchange data and functionality easily and securely. It is a set of defined rules that explain how applications communicate with one another. APIs sit between an application and web server, acting as an intermediary layer that processes data transfer between systems.

API is a backbone for any application from websites to mobile apps, we need a backend that will help us to structure our application. But if that backbone is not functional the whole application will fall apart.

### 1.1.1 Here's how an API work :

- A. A client application initiates an API call** to retrieve information - also known as a *request*. This request is processed from an application to the web server via the APIs Uniform Resource Identifier (URI) and includes a request verb, headers, and sometimes a request body.
- B. After receiving a valid request** , the API makes a call to the external program or web server.
- C. The server sends a *response*** to the API with the requested information.
- D. The API transfers the data** to the initial requesting application.

APIs are used everywhere , Signing up and Logging in to any application there is an API call. When we google something , we type and hit enter - all links are displayed, it is all because of API. It is a backbone to any application.

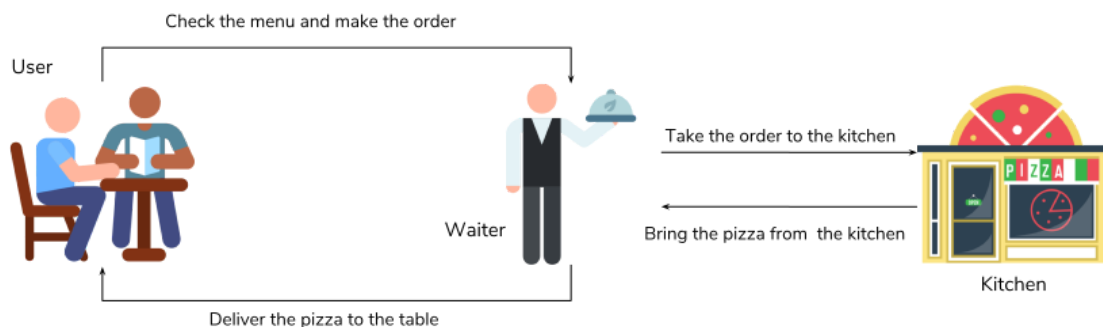
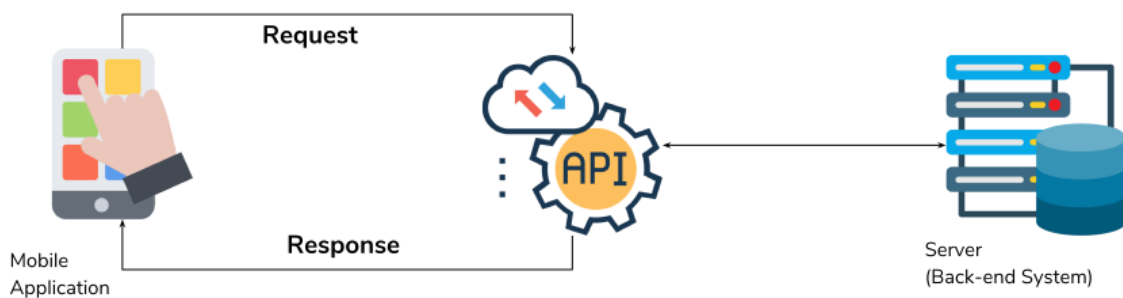


Fig - 1.1 Working of API

Developers develop the API, but *How do they test them? How can they share the APIs among the team members? How can they maintain and document them ?*

### 1.1.1 Problem faced in current tools :

There are many tools to solve those problems but are they really solving the problem, or giving extra work to do for developers.

- Mostly every application uses API for communication between user and database. APIs are an important part that need to be maintained, tested and documented. **How to maintain APIs? How to Test them? And how can we Document them?**
- Many of the tools that provide a platform to document are **static in nature**. Everytime users have to **manually add the APIs requests** and provide details to those specific requests.
- Few tools are **just for documenting only, no testing, no development**. (*platform like Swagger*)
- A single project is developed by a team of 10-20 developers at a time, in which 5-6 of them will be working on building the APIs. How they can work on the same API and not produce any conflicts. That's the problem with current tools. **There is no Collaborating feature to them.**



## 1.2 Proposed Solution

The Purpose of “**Collaborative API Software : Alter-API**” is to provide a development environment for the developers to test, develop, document and collaborate with others efficiently and conveniently.

- A. It is an API platform for building and testing APIs. It simplifies each step of the **API lifecycle and streamlines collaboration** so developers can create better APIs - faster. Unlike other platforms , it provides more features to work on.
- B. It gives the ability to create **different workspaces for different projects**. Users can have workspaces respective to them or the team, within workspace there are collections helping developers to divide the APIs into different categories and within a single collection , they can simulate many requests.
- C. Quickly and easily requests directly within Alter-API. **Automate manual test** and Communicate the expected behaviour of an API by simulating endpoints. Simulating requests with ease and provides users with all the options needed for it.
- D. It helps in documenting your APIs that can be shared and downloaded among the team members. Unlike other tools, in which users have to manually add the documentation for each request they simulate.  
But in Alter-API, users don't need to make documentation manually, they need to just simulate requests, **documentation will be automatically** generated with respective headers, body and response from API.  
Users have flexibility to rearrange each request document in any order. Just drag and drop.
- E. Users can create their **own API repositories and invite other developers** who can collaborate with each other and can raise comments if there is an issue. Best thing about Alter-API is that users can invite team members and work together with them, no need to worry about the conflicts.  
If there is an issue , just raise a comment within a particular request and resolve it in minutes.

- F. It provides more **flexibility** than the currently available tools / platforms. Providing all features into one package.

## 1.3 Project Scope

There is always a communication gap between the different teams during a software development process because everyone is not well versed with all aspects of development.

This software removes the gap between the frontend and backend developers. It helps in collaborative work experience and helping each other at each step.

It is a convenient, efficient, free tool for smooth and fast API development.

This software solves the problem for developers to test their API's while developing them and publishing them to the server. They can also document the API and their endpoints for other team members and future use. New members don't have to nag the developers, What data to pass? How does it work?

## 1.4 Features and Functionalities

- **Simulating APIs** : Automate manual tests and Communicate the expected behaviour of an API by simulating endpoints. Users can test and develop APIs. All the necessary options are provided to simulate requests.
- **Saving Requests** : Users can save API endpoints with their respective response requests. These requests will be visible to the team and they can also work on it at the same time.
- **Login / Signup** : Users can make their own account and have personalised sessions specific to a user only.
- **Creating Workspace** : Users have flexibility to create different workspace for different projects and can invite the team members having the collaborative session.
- **Inviting Team** : Users can invite the team members having the collaborative session. Different roles can be assigned to invited members as per the project requirements. They can raise issues and resolve on the platform only just by commenting on the requests.
- **CRUD Operations** : CREATE, READ, UPDATE and DELETE these operations will be used throughout the whole application for Logging user in, Saving requests, Creating workspaces etc.
- **Documenting API** : Users have power to create documentation out of their saved endpoints, responses and many more things. Documentation will be generated automatically based on saved requests and users can also customise based on their requirements.
- **Sharing APIs** : Users can share among other members and download the APIs for future use. Sharing helps in being more productive , developers don't need to ask for updates , they can monitor at the same time as APIs are being developed.

## **Chapter 2**

### **PROJECT DESCRIPTION**

2.1 Methodology Used

2.2 Technology Used

2.3 Required Tools

2.4 System Requirements

*What project is it about?* Project titled “**Collaborative API Software : Alter-API**” is a platform providing tools to developers for simulating, testing , monitoring and documenting APIs with a collaborative experience.

*Let's break down all the terms mentioned.*

**API** ( Application Programming Interface ) is just a way of communicating between two applications. It enables applications to exchange data and functionality easily and securely. It is a set of defined rules that explain how applications communicate with one another. APIs sit between an application and web server, acting as an intermediary layer that processes data transfer between systems.

### **Simulating API**

Developers build the APIs using preferred framework but now question arises where they are going to test or simulate them. The only way is by integrating it to the Frontend but that's a costly task and time consuming. Developers cannot integrate it every time there is a change in API.

Developers need a different way to test the APIs. Now the project comes into play, developers don't need to integrate API to frontend every time, they can just create a new request and type in the URI and run those requests as they want.

The Application will save time and money and help developers to be more productive.

### **Documenting API**

APIs are integrated right, then how developers will identify which API needs to be used at what component of the application? and what data needs to be passed? What type of response will be received ? . Answer to the above questions is *Documentation*.

Developers need documentation for the APIs for integrating them. Documentation will contain all the details required.

### **Collaborating Experience**

An important part of a software engineer's job is to gather, understand, and finalise requirements. This often involves multiple meetings with multiple stakeholders (customer, security/network team, managers).

In Alter-API developers can work together with other team members on the same workspace with no conflicts and can also monitor the whole project.

## 2.1 Methodology Used

This project uses *Agile Methodology*, meaning a practice that promotes continuous iteration of development and testing throughout the software development life cycle of the project. In the Agile model in software testing, both development and testing activities are concurrent, unlike the Waterfall model.

As the application is being made by “Deepak Sharma” single person, the process of developing application was divided into cycles. With each cycle a single feature of the application was developed and tested.

Agile emphasises collaboration, flexibility, continuous improvement, and high quality results. It aims to be clear and measurable by using six main “deliverables” to track progress and create the product.

Basically there are two most popular Agile Development Life Cycle SCRUM & KANBAN.

### 2.1.1 Phases of Agile Methodology :

#### 1. Planning :

- a. Identifying the requirements for the current cycle.
- b. Gathering information related to the project.
- c. Service Level Agreements and its conditions.

#### 2. Analysis :

- a. Capturing of user stories.
- b. Prioritising the stories.
- c. Defining the iteration span / time.
- d. Resource planning for development.

#### 3. Design :

- a. Break down of tasks.
- b. Test Scenario preparation for each task.
- c. Regression Automation Framework.

#### 4. Execution :

- a. Coding

- b. Unit Testing
- c. Execution of manual test scenarios
- d. Defect report.

## 5. Wrapping :

- a. Small releases
- b. Demos and reviews.
- c. Develop new stories based on the need.
- d. Process improvement based on end of iteration review comments.

## 6. Closure :

- a. Deployment of the current application
- b. Production Support.

This project development uses KANBAN, because it is easy to use as an individual developer on a project.

### 2.1.2 KANBAN

Kanban is a popular framework used to implement agile development. It requires real time communication of capacity and full transparency of work. Work items represented visually on a *Kanban board*, allowing team members to see the state of every piece of work at any time.

#### ***How was KANBAN used in the project ?***

I used a tool called “**Trello**”, which is an Atlassian product to manage the project.

Trello provides a kanban board that contains multiple cards, which are divided into different stages of the application. Each card contains a user story of a certain feature and sub tasks are listed below it. They have their own status *Todo, Progress, Halt, Done*.

- a. I divide my project into different modules and those modules are further divided into components.
- b. For each component I made a card and listed all tasks to be done under it.
- c. Newly created card is assigned with *Todo* status as the development of that component starts, the status changes to progress.

- d. This process goes on till the component is ready to deploy.
- e. We can track the progress of the application and improve the development process.
- f. After each iteration I added new cards as required.

**Here is how my Trello Board looks like :**

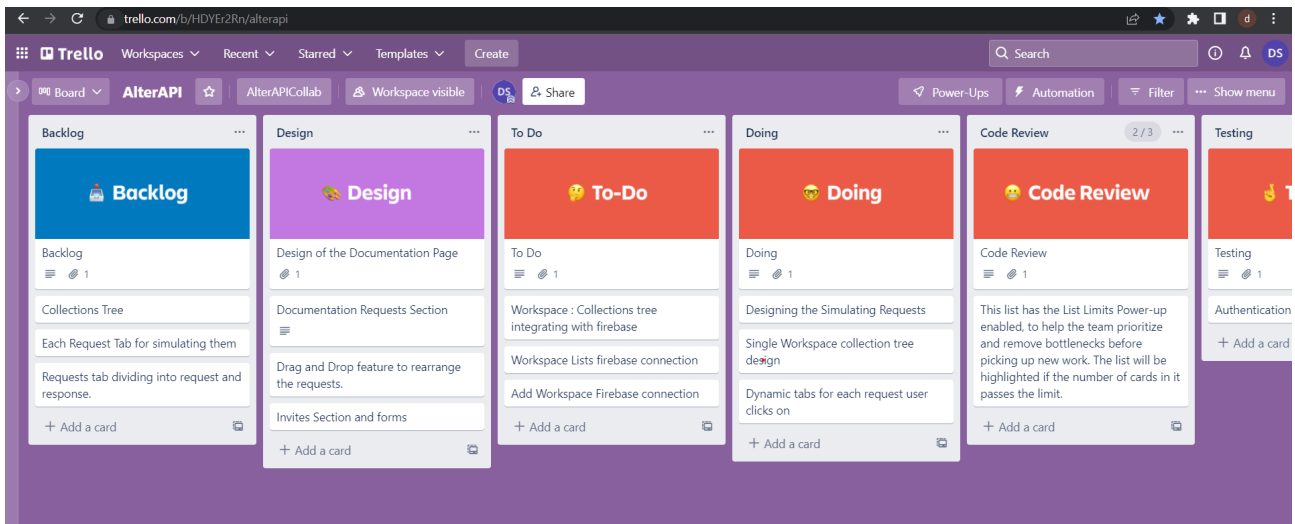


Fig 2.1 Trello Kanban (a)

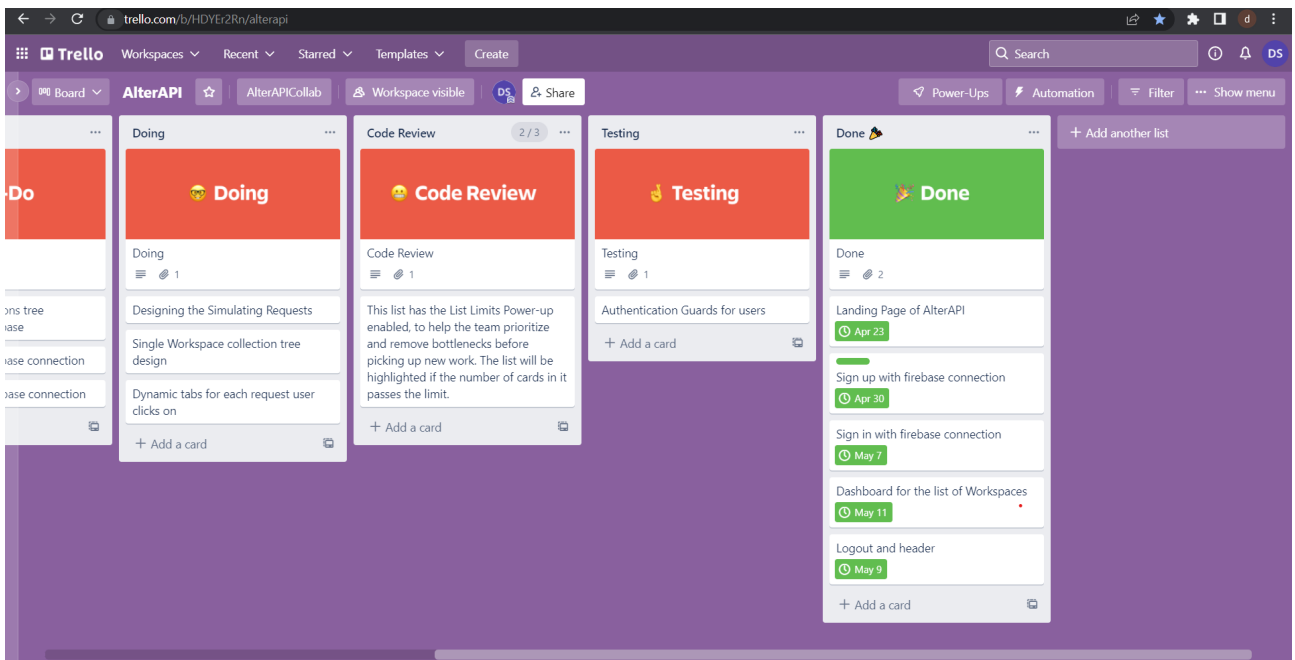


Fig 2.2 Trello Kanban (b)



## 2.2 Technology Used

**Software technology** A general term covering the development methods, programming languages, and tools to support them that may be used in the development of software.

This project “Collaborative API Software : Alter-API” is a web application which consists of different parts: *Frontend, Backend, Database, Web Server*.

### 2.2.1 Frontend

The Front-end part of the application is build using following tech stack :

#### Angular 13

It is a TypeScript Framework used to build single page applications, also help in creating modules and components.

Angular is a TypeScript-based free and open-source web application framework led by the Angular Team at Google and by a community of individuals and corporations. Angular is a complete rewrite from the same team that built AngularJS.

#### *Why Angular ?*

JavaScript is the most commonly used client-side scripting language. It is written into HTML documents to enable interactions with web pages in many unique ways. As a relatively easy-to-learn language with pervasive support, it is well-suited to develop modern applications.

But is JavaScript ideal for developing single-page applications that require modularity, testability, and developer productivity? Perhaps not.

These days, we have a variety of frameworks and libraries designed to provide alternative solutions. With respect to front-end web development, Angular addresses many, if not all, of the issues developers face when using JavaScript on its own.

### *How to Build an Angular Project?*

Step 1. `npm install -g @angular/cli`

Step 2. `ng new [PROJECT NAME]`

Step 3. `cd [PROJECT NAME]`

`ng serve`

## **HTML (Hyper Text Markup Language)**

It is the basic building block of any web app, used for structuring the app.

## **CSS ( Cascading Style Sheet )**

It is used for styling the app. Used SCSS also.

## **TypeScript**

JavaScript was introduced as a language for the client side. The development of Node.js has marked JavaScript as an emerging server-side technology too. However, as JavaScript code grows, it tends to get messier, making it difficult to maintain and reuse the code. Moreover, its failure to embrace the features of Object Orientation, strong type checking and compile-time error checks prevents JavaScript from succeeding at the enterprise level as a full-fledged server-side technology.

TypeScript is a strongly typed, object oriented, compiled language. It was designed by Anders Hejlsberg (designer of C#) at Microsoft. TypeScript is both a language and a set of tools. TypeScript is a typed superset of JavaScript compiled to JavaScript.

## **Tailwind**

An API for your design system. Utility classes help you work within the constraints of a system instead of littering your stylesheets with arbitrary values. They make it easy to be consistent with colour choices, spacing, typography, shadows, and everything else that makes up a well-engineered design system.

## **Angular Material**

Angular Material is a UI component library for Angular developers. Angular Material components help in constructing attractive, consistent, and functional web pages and web applications while adhering to modern web design principles like browser portability, device independence, and graceful degradation. It helps in creating faster, beautiful, and responsive websites. It is inspired by Google Material Design.



**Angular**



**TypeScript**



**Tailwind**

Fig 2.3 Frontend Technologies

### **2.2.2 Backend**

back-end development is all about making these apps render server-side. But it's a bit more involved than that. While the previous statement holds true, back-end developers also create services that process business logic and access other resources such as databases, file servers, cloud services and more. These services are the backbone of any application and can be accessed and used not only by server-side rendering apps but also from client-side rendering apps.

The Back-end part of the application is build using few technology only, as follows :

### **Nodejs**

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

### **Express**

Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications.



Fig 2.4 Backend Technologies

### **2.2.3 Database**

This project is based on collaborative experience so, it uses following Database :

**Firestore & Firebase :** Firebase is an app development platform that helps you build and grow apps. It also provides a real time database called *Firestore*.

Cloud Firestore is a cloud-hosted, NoSQL database that your Apple, Android, and web apps can access directly via native SDKs. Cloud Firestore is also available in native Node.js, Java, Python, Unity, C++ and Go SDKs, in addition to REST and RPC APIs.

## NoSQL Database

NoSQL databases (aka "not only SQL") are non-tabular databases and store data differently than relational tables. NoSQL databases come in a variety of types based on their data model. The main types are document, key-value, wide-column, and graph. They provide flexible schemas and scale easily with large amounts of data and high user loads.



Fig 2.5 NoSQL Database technology

### 2.2.4 Service for Hosting

**Firestore** is built for the modern web developer. Websites and apps are more powerful than ever with the rise of front-end JavaScript frameworks like Angular and static generator tools like Jekyll. Whether you are deploying a simple app landing page or a complex Progressive Web App (PWA), Hosting gives you the infrastructure, features, and tooling tailored to deploying and managing websites and apps.



Firestore Hosting

Fig 2.6 Firestore Hosting

## 2.3 Required Tools

There are different tools that can be used for developing an application.

Here is the list of tools this project uses while building it :

- a. **Visual Studio Code** : It is an IDE for writing code for the application.
- b. **Git & GitHub** : *Git* is used for version controlling the project and pushing the code to *GitHub*. It was also used for tracking the project progress.
- c. **Chrome Dev Tools** : It is used for bug fixing and testing the application.
- d. **Figma** : It is used for building the visuals / screens for different modules and illustration for the app.
- e. **Trello** : It is used for tracking the progress of the project.



VS Code



GitHub



Crome



Figma



Trello

Fig 2.1 Tools used for developing application

## **2.4 System Requirements**

### **1. Developer requirements**

- Laptop or PC
- Active Internet Connection

#### **1.1 Hardware requirements**

- Processor : at least i3 processor or higher
- RAM : 4GB min. (8GB recommended)
- Hard Disk : 4 GB min.

#### **1.2 Software requirements**

- OS : Windows 7 or higher/Linux/MacOS
- VS Code
- NodeJS Server
- Web browser

### **2. Client-side/User requirements**

- Desktop,Laptop, Phone or Tablet
- Mac OS / Windows

### **3. Database Requirements**

- NO SQL , Firebase.

# **Chapter 3**

## **System Analysis**

3.1 Data Flow Diagram

3.2 Use Case Description and Diagram

3.3 ER Diagram

3.4 Class Diagram

3.5 State Transition Diagram



System Analysis is a process of identifying the problems and dividing them into different components. Its purpose is to study the system and what the objectives are.  
*Let's see what the system does.*

### 3.1 Data Flow Diagram

Data Flow Diagram (DFD) represents the flow of the data through the system.

#### 3.1.1 Level 0 DFD

Processes	Entities	Data Store
Alter-API	User	<i>None</i>

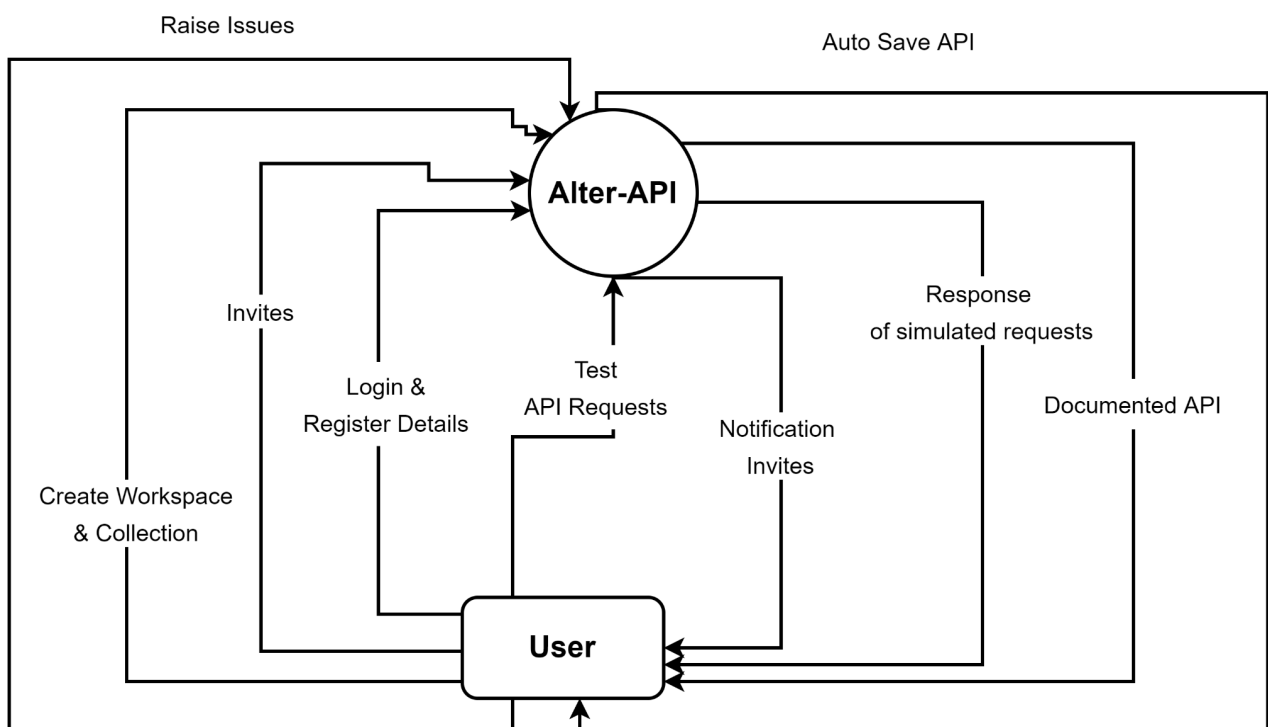


Fig 3.1 Level 0 DFD

### 3.1.2 Level 1 DFD

Processes	Entities	Data Store
Register Login Create Workspace Test Request Notification Documentation Raise Issues	Admin ( User ) Developer ( User )	User Store Workspace Store Request Store

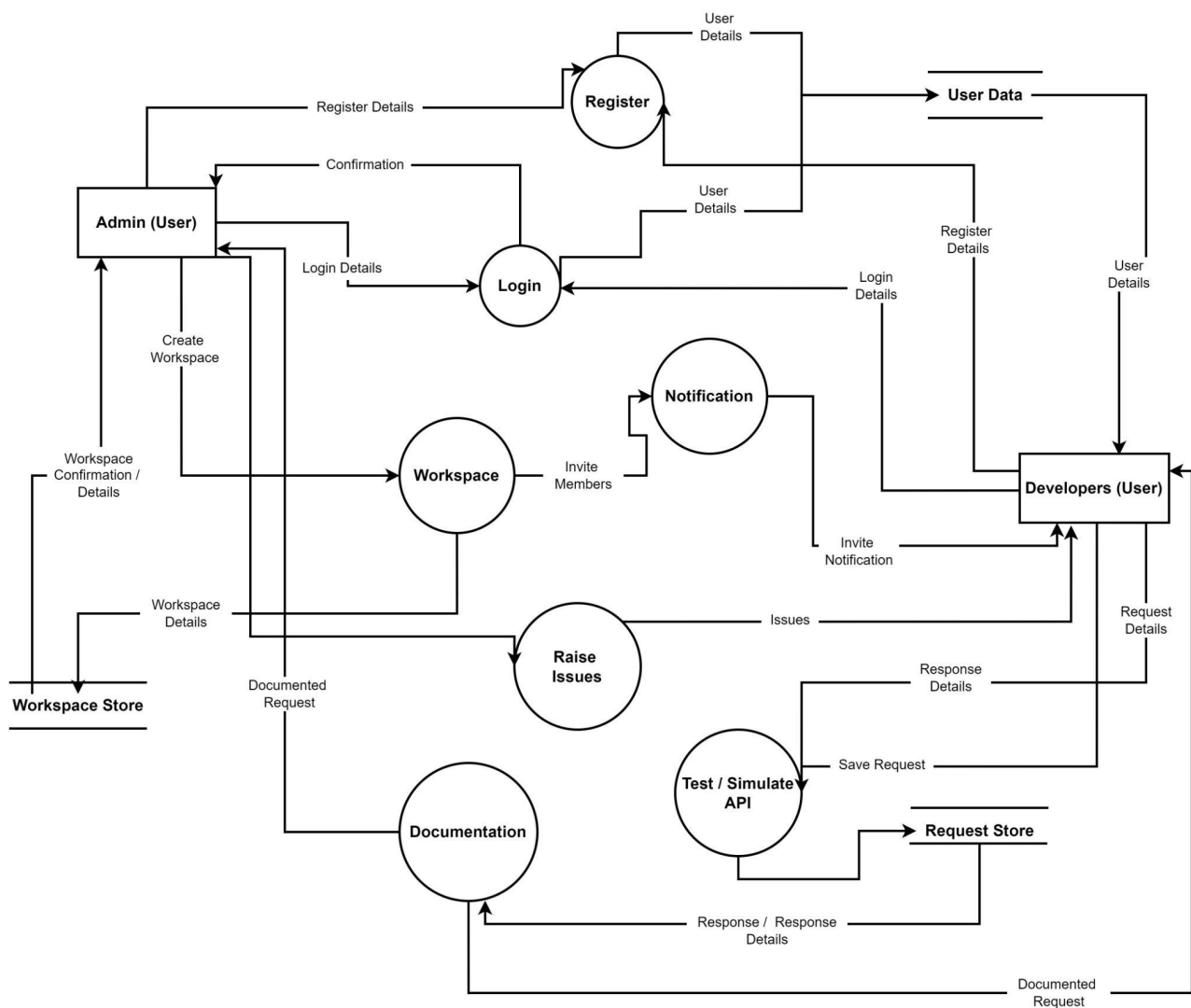


Fig 3.2 Level 1 DFD

### 3.1.2 Level 2 DFD

#### User Authentication :

Processes	Entities	Data Store
Register Login	User	User Store

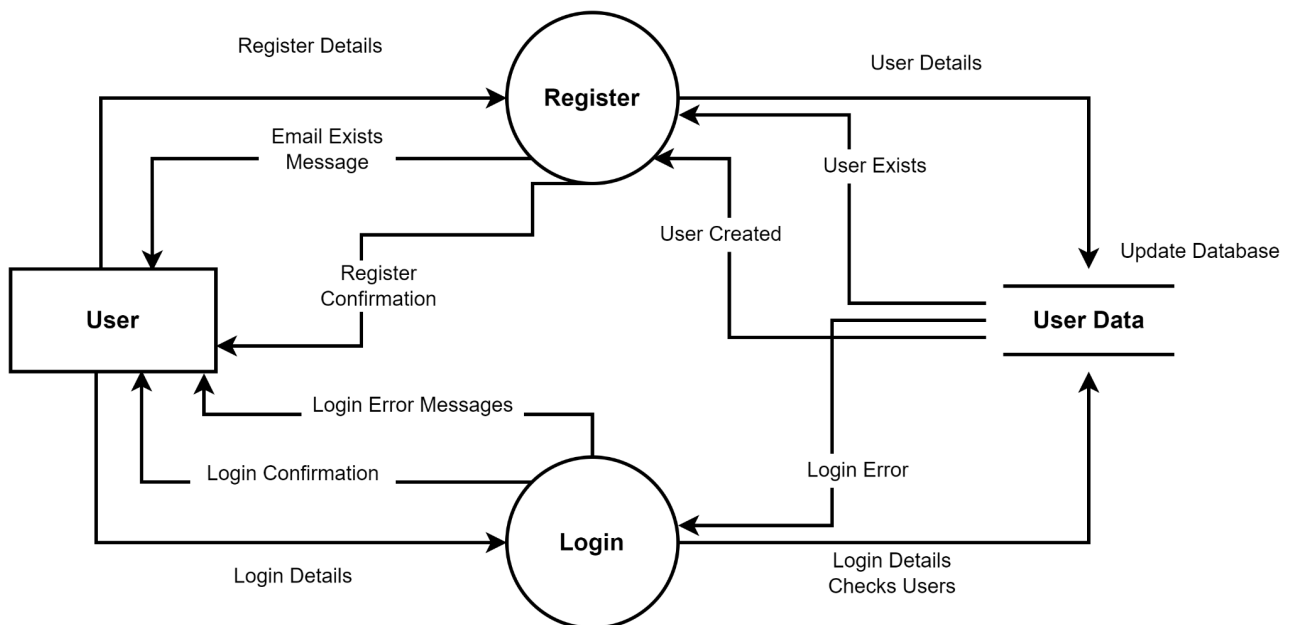


Fig 3.3 Level 2 DFD - User Authentication

## Workspace :

Processes	Entities	Data Store
Create Workspace Create Collection Add API / Collection Invites	User	User Store Workspace Store Request Store

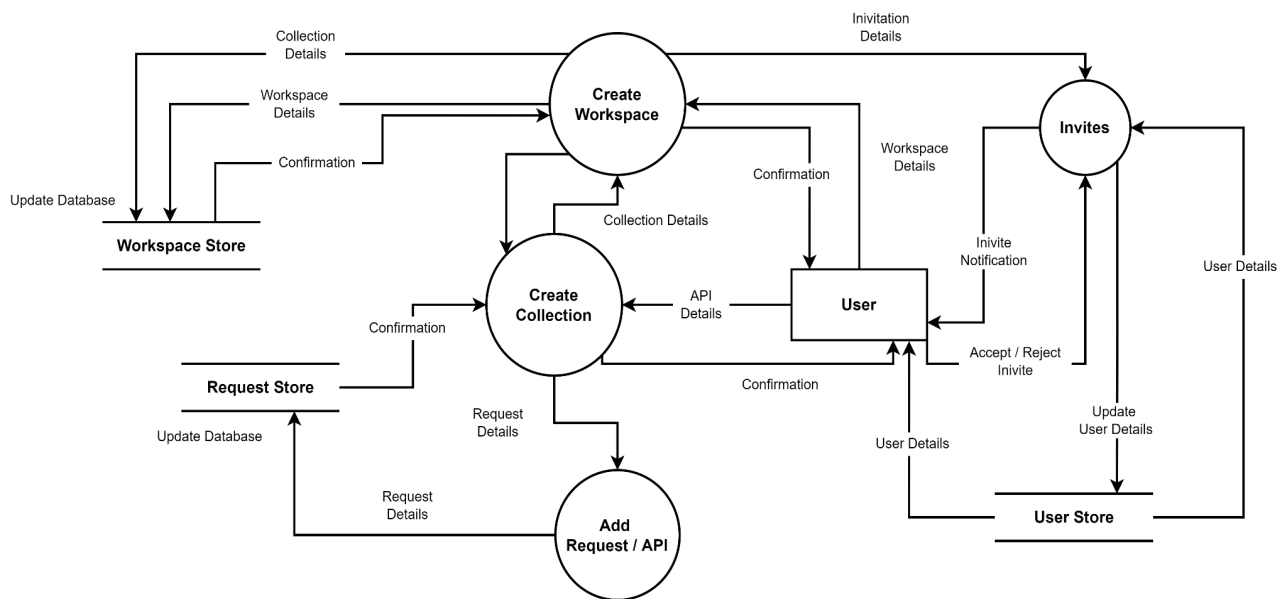


Fig 3.4 Level 2 DFD Workspace

## 3.2 Use Case Description

Use Case Description is written describing how a user will interact with the system or application. It outlines, from a user's point of view, a system's behaviour as it responds to a request. Each use case is represented as a sequence of simple steps, beginning with the user's goal and ending when that goal is fulfilled.

### 3.2.1

Use Case 01	Login / Register User
Objective	How user can login / register into application
Actors	Developers
Pre-Condition	Should have login credentials
Post-Condition	Application is accessible to the logged in users
Basic Flow	<ol style="list-style-type: none"><li>1. Actors will enter the login / register credentials.</li><li>2. Passwords and other credentials are validated.</li><li>3. Actors will be redirected to their dashboard.</li></ol>
Alternative Flow	<ol style="list-style-type: none"><li>1. An error message will be displayed if the Actor is not authorised.</li><li>2. Invalid or expired credentials.</li></ol>

### 3.2.2

Use Case 02	Creating Workspace and Adding requests.
Objective	How users can create their own workspace and add requests to test them.
Actors	Developers
Pre-Condition	None or previous workspaces are listed on the dashboard.
Post-Condition	A new Workspace will be added and perform action under it.
Basic Flow	<ol style="list-style-type: none"><li>1. Actor will click on “<i>new workspace</i>” and fill in the details.</li><li>2. Actors could add team members' email and invite them.</li><li>3. After clicking on the submit button a new workspace is being created.</li><li>4. Now actors have access to the whole workspace and add collections, then add a new <i>request</i> to test it.</li></ol>
Alternative Flow	<ol style="list-style-type: none"><li>1. Error will be displayed in case of invalid details.</li><li>2. An error message will be displayed if the Actor is not authorised.</li></ol>

### 3.2.3

Use Case 03	Simulating Request
Objective	How users can build, test and save requests
Actors	Developers
Pre-Condition	Empty workspace with no APIs / requests.
Post-Condition	New requests will be added and autosaved for documentation.
Basic Flow	<ol style="list-style-type: none"><li>1. Click on the add button and a new request tab is added and actors get to add / type URI of the API.</li><li>2. Actors can add other additional details to test requests.</li><li>3. After sending the request, the response is displayed and it will be saved in DB.</li></ol>
Alternative Flow	<ol style="list-style-type: none"><li>1. Error will be displayed in case of invalid details.</li><li>2. An error message will be displayed if the Actor is not authorised.</li></ol>

## Use Case Diagram

Use Case Diagrams model the behaviour of a system and help to capture the requirements of the system. They describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors.

The use cases and actors in the use-case diagram describe what the system does and how the actors use it.

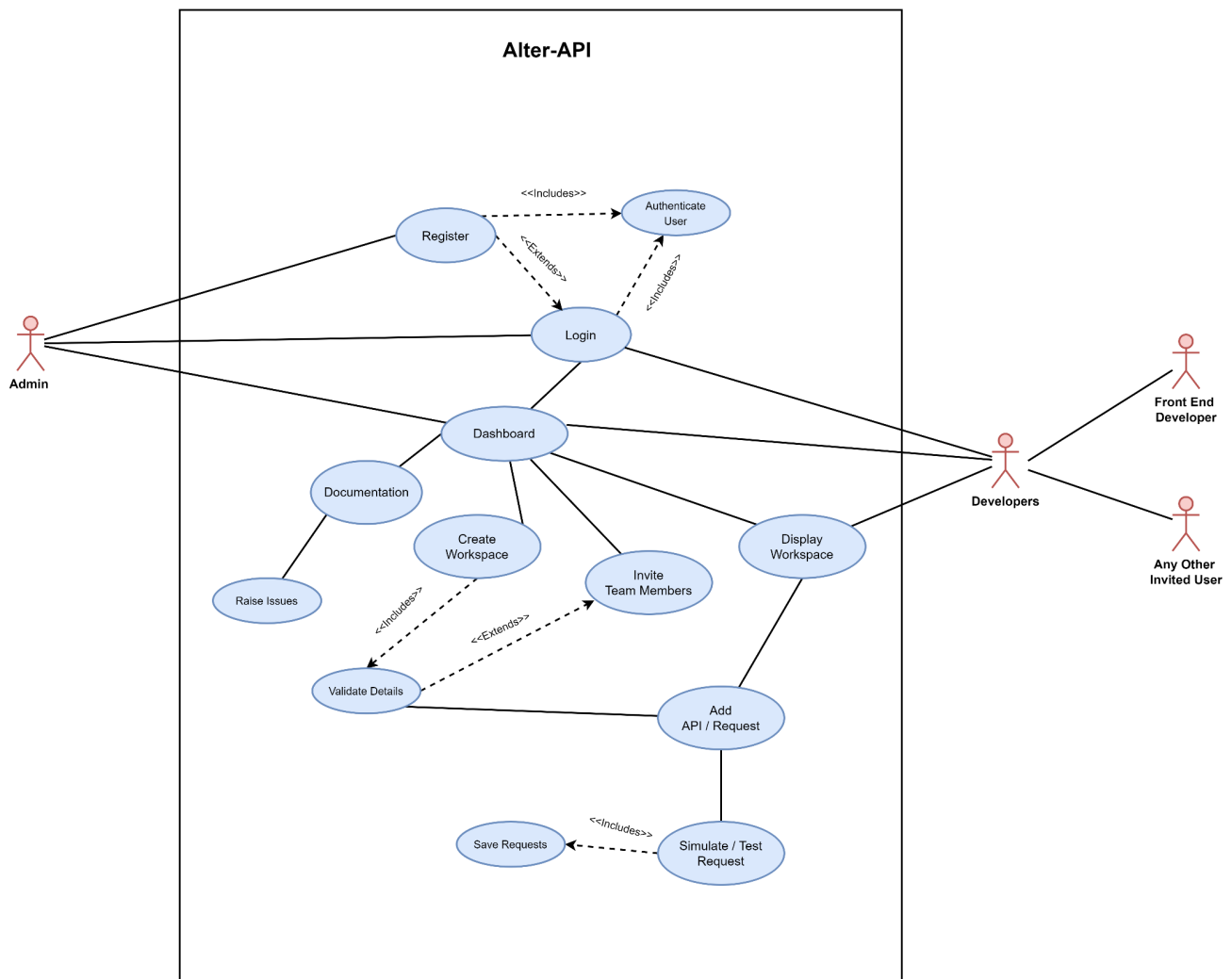


Fig 3.5 Use Case Diagram



### 3.3 Entity Relationship Diagram

An Entity Relationship Diagram ( ERD ) is a visual representation of different entities within a system and how they are related to each other.

ERDs are widely used to design relational databases. The entities in the ER become tables, attributes and lastly converted into database schema. ERDs help in visualising all type of relationship between the entities.

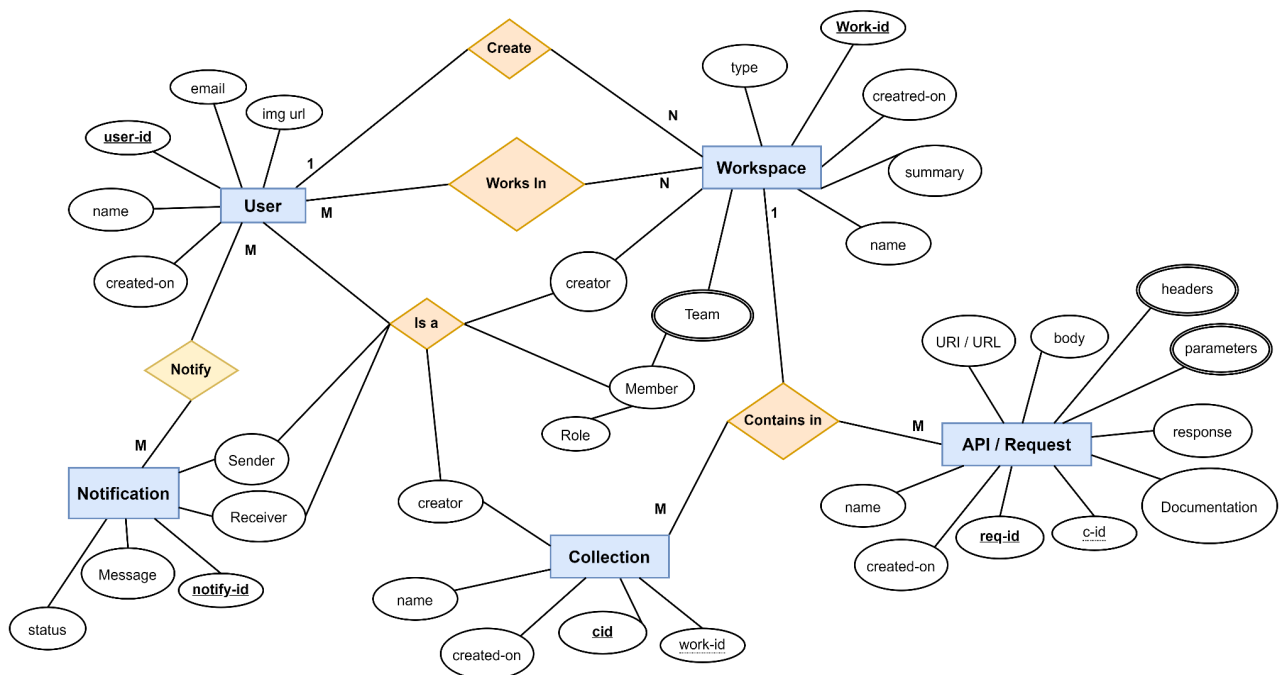


Fig 3.6 Entity Relationship Diagram

### 3.4 Class Diagram

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them. A class consists of its object , and also it may inherit from other classes.

A class diagram is used to visualise, describe, document various different aspects of the system.

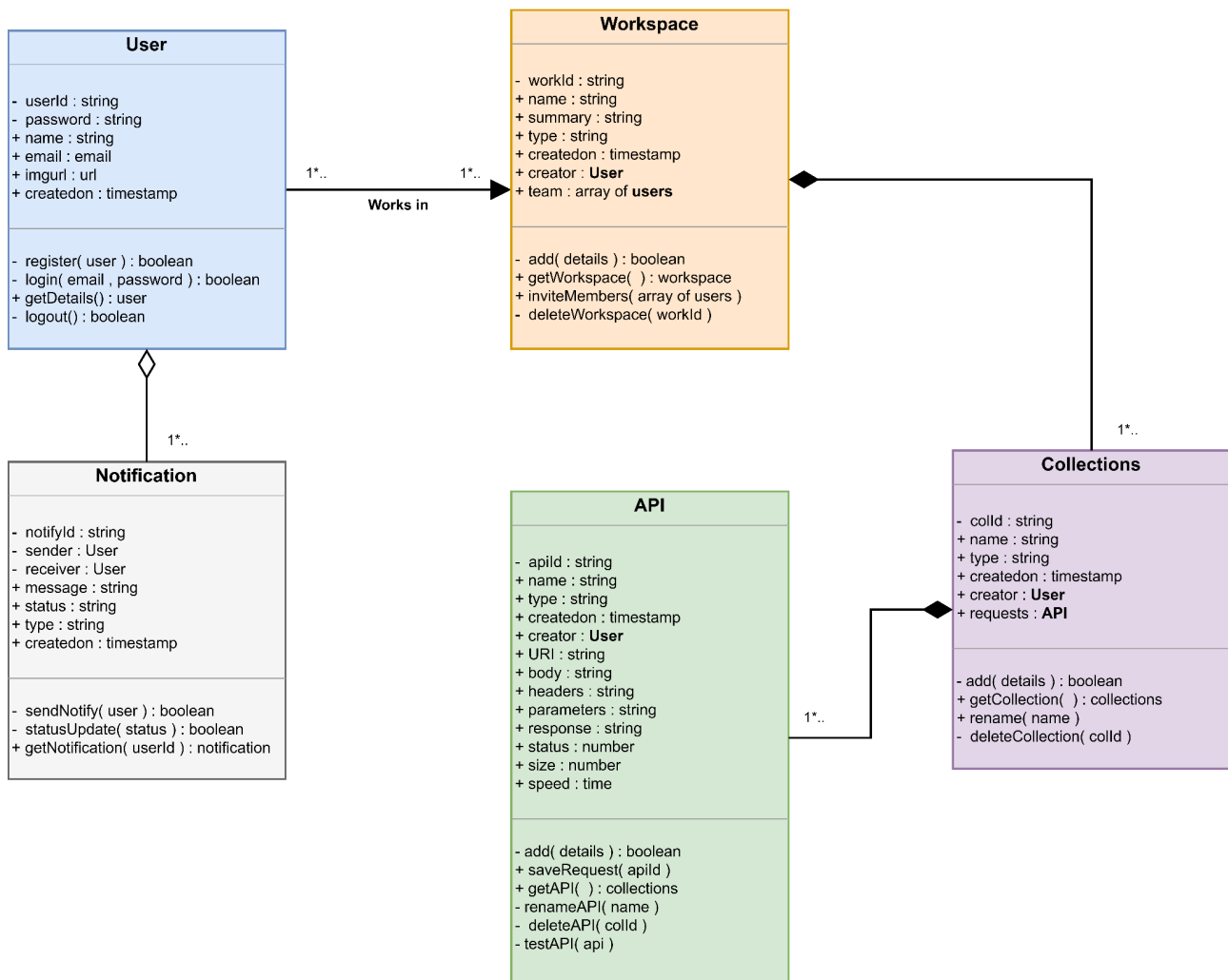


Fig 3.7 Class Diagram

### 3.5 State Transition Diagram

A state transition diagram is a graphical representation of a state machine. It shows a behavioural model consisting of states, transitions and actions, as well as the events that affect these.

State diagram helps in visualising the progress of event driven objects in a reactive system.

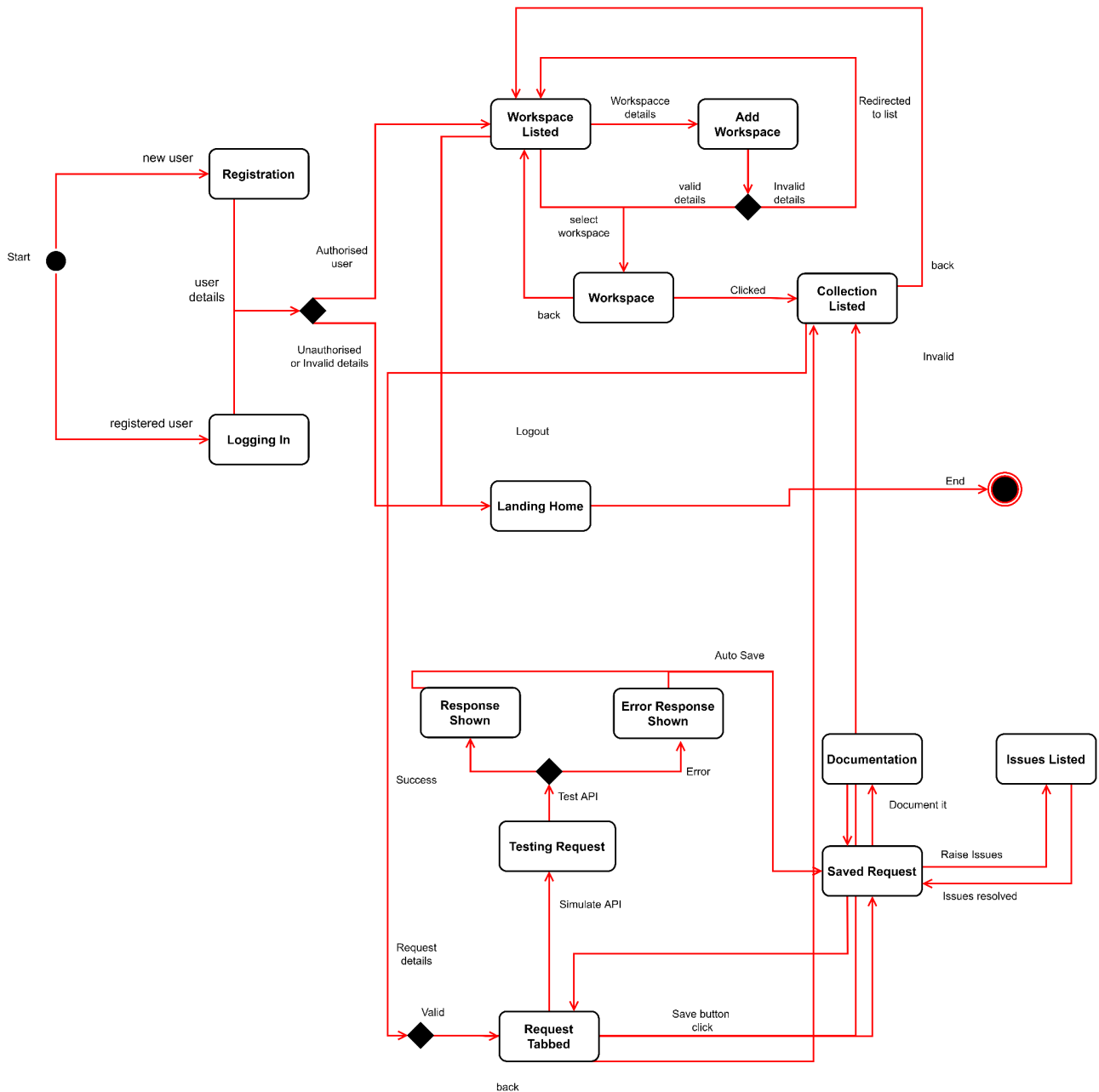


Fig 3.8 State Transition Diagram

# **Chapter 4**

## **System Design**

4.1 Database Design

4.2 Project Structure

4.3 Form Design

4.4 UI Design

## 4.1 Database Design

This project uses a **NoSQL** database due to its requirements and tools used for the application.

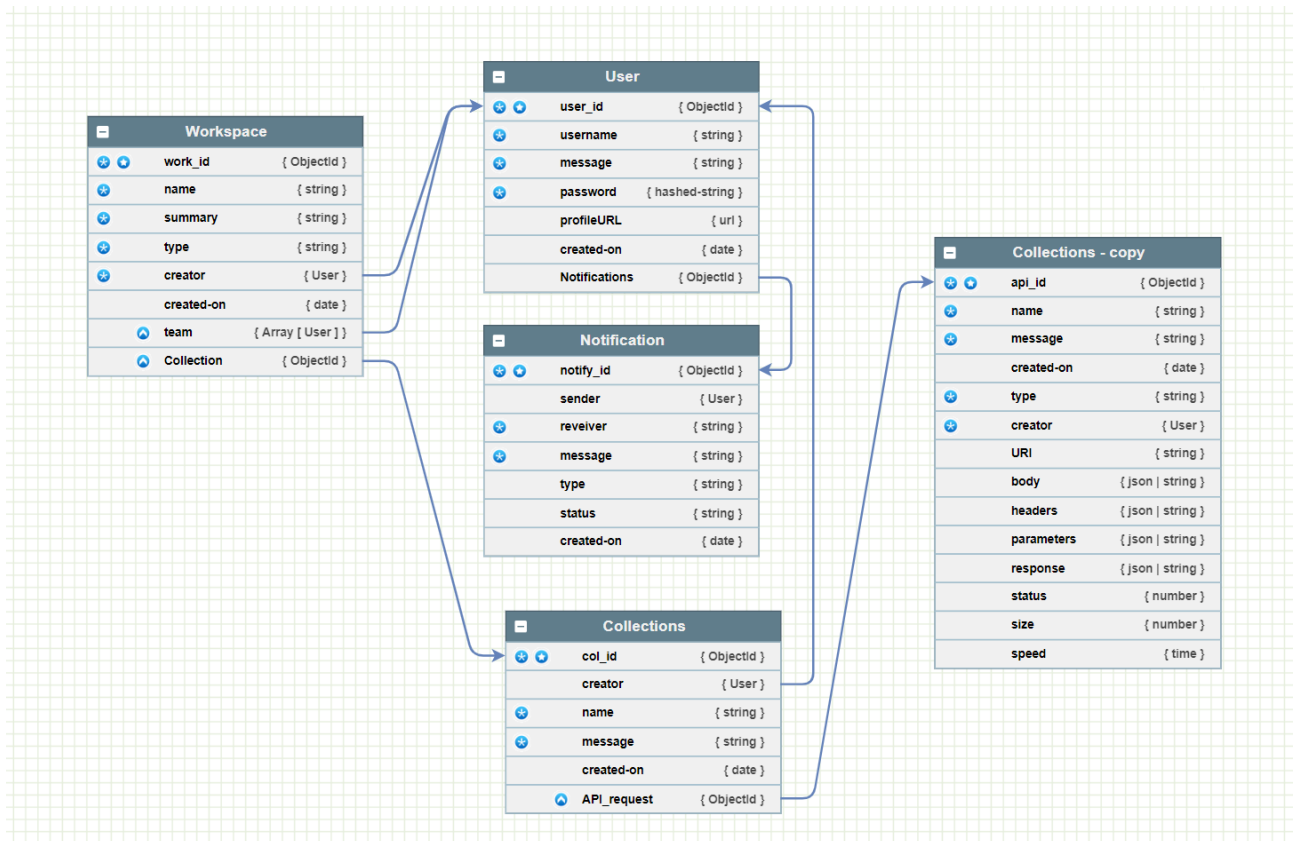


Fig 4.1 Database Design

## 4.2 Project Structure

This project uses **Angular** for developing the application, and using **VS Code** as an IDE.

Let's start with the base of the application. How to create a angular project :

`ng new [Project-Name]` : for creating a new project

`ng serve` : for serving the project on localhost

It is powered by the packages that've been installed in it and that are being configured by the `package.json` file. Let's have a look at it :



```
{ package.json X
package.json > {} devDependencies
1  {
2    "name": "alter-api-collab",
3    "version": "0.0.0",
4    > Debug
5    "scripts": {
6      "ng": "ng",
7      "start": "ng serve",
8      "build": "ng build",
9      "watch": "ng build --watch --configuration development",
10     "test": "ng test"
11   },
12   "private": true,
13   "dependencies": {
14     "@angular/animations": "~13.3.0",
15     "@angular/cdk": "^13.3.4",
16     "@angular/common": "~13.3.0",
17     "@angular/compiler": "~13.3.0",
18     "@angular/core": "~13.3.0",
19     "@angular/fire": "^7.3.0",
20     "@angular/forms": "~13.3.0",
21     "@angular/material": "^13.3.4",
22     "@angular/platform-browser": "~13.3.0",
23     "@angular/platform-browser-dynamic": "~13.3.0",
24     "@angular/router": "~13.3.0",
25     "rxjs": "~7.5.0",
26     "t-writer.js": "^1.0.4",
27     "tslib": "^2.3.0",
28     "zone.js": "~0.11.4"
29   },
30   "devDependencies": {
31     "@angular-devkit/build-angular": "~13.3.3",
32     "@angular/cli": "~13.3.3",
33     "@angular/compiler-cli": "~13.3.0",
34     "@types/jasmine": "~3.10.0",
35     "@types/node": "^12.11.1",
36     "autoprefixer": "^10.4.5",
37     "jasmine-core": "~4.0.0",
38     "karma": "~6.3.0",
```

Fig 4.2 Package.json file ( Dependencies )

In Angular we divide our application into different components that can be reused anywhere. First look at the overall project structure :

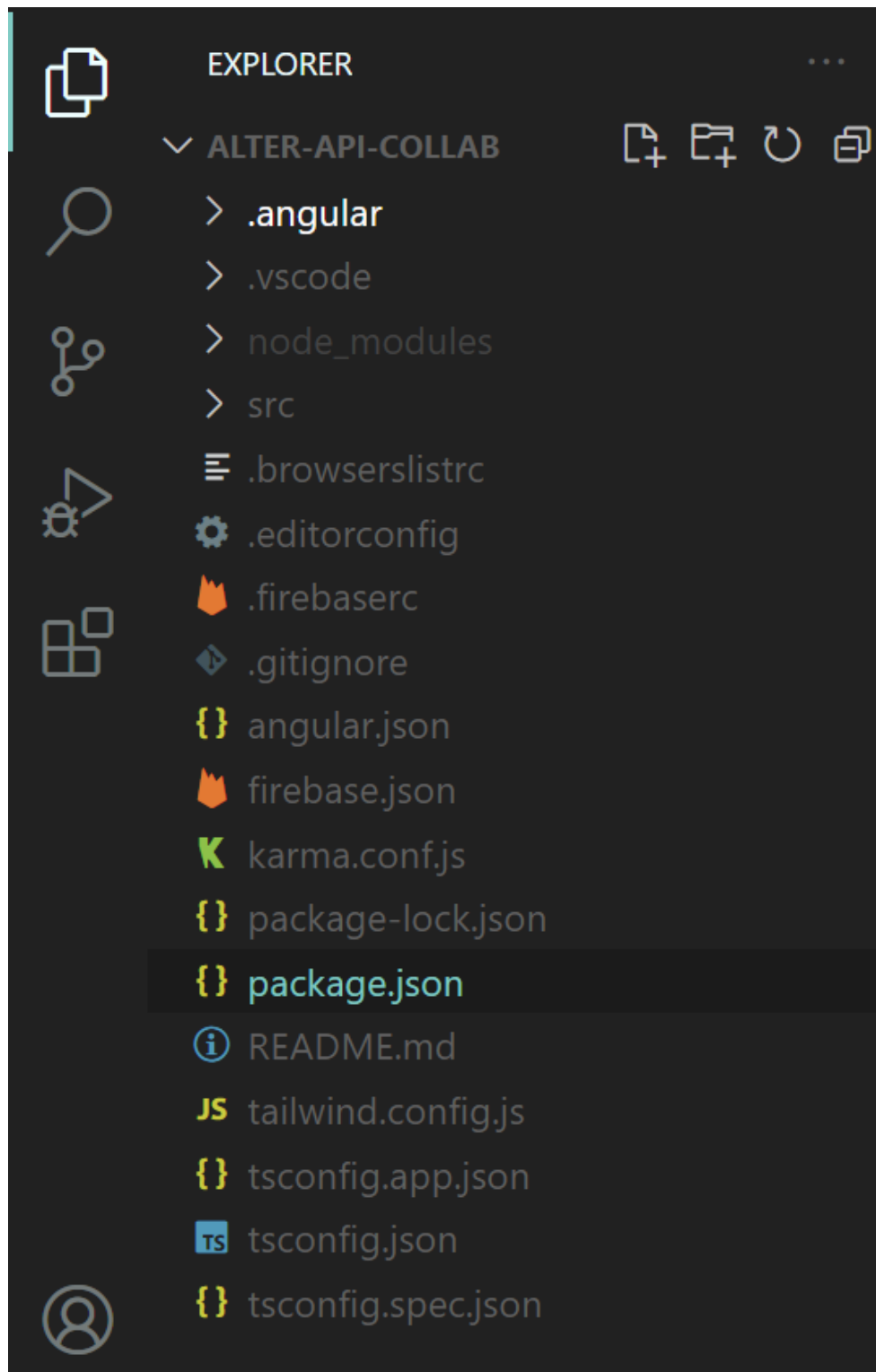


Fig 4.3 Project Structure - app folder

Now let's look deep into how we structure our application in the form of components and modules.

```
-app
  -shared
    -services
    -pipes
    -components
    -models

  -modules
    -users
      users.component.ts
      users.component.html
      users.module.ts
      users.route.ts

    -organisations
      organisations.component.ts
      organisations.component.html
      organisations.module.ts
      organisations.route.ts

  -app.component.ts
  -app.component.html
  -app.module.ts
  -app-routing.module.ts
```

Fig 4.4 File Structure



## Project Structure :

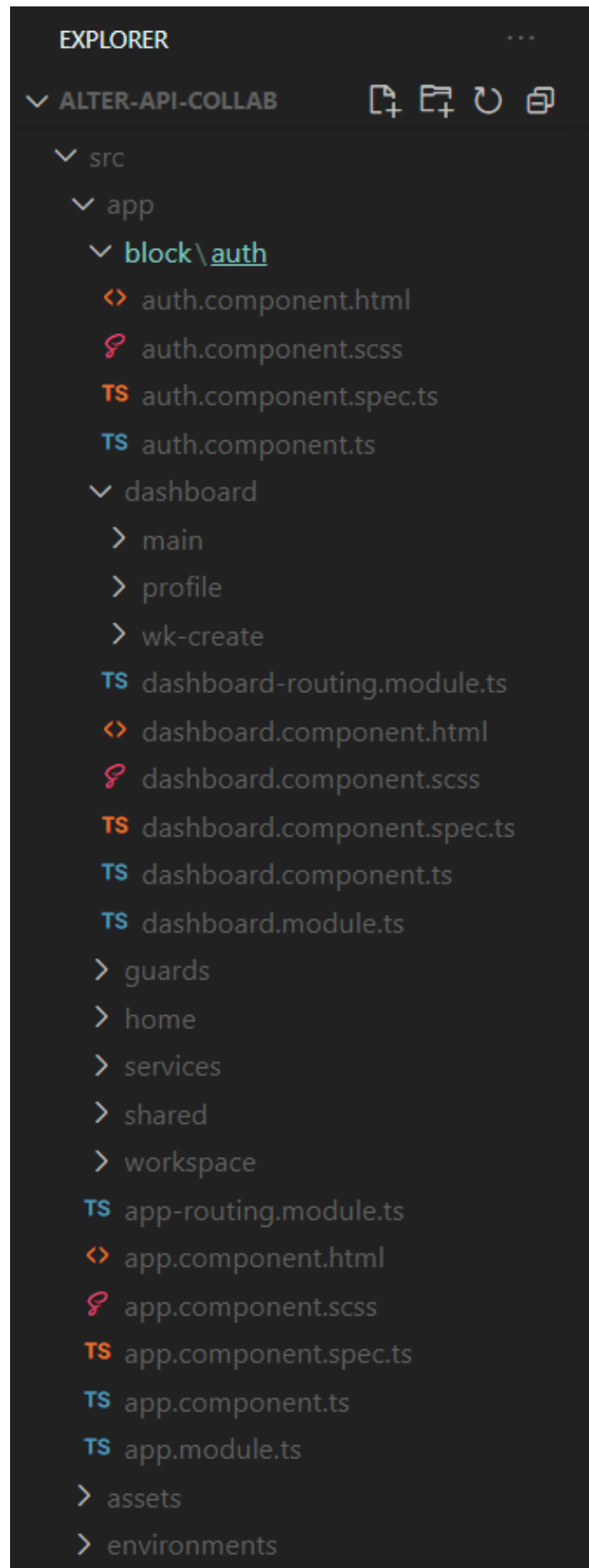


Fig 4.5 Project Structure

## 4.3 Coding

The project uses vs code to code the application. List of languages used for coding :

1. HTML
2. SCSS
3. TypeScript and JavaScript

**Components** are build is divide into three section

- a. Template (view) : It contains the HTML code.
- b. Styling : It contains SCSS code.
- c. Business Logic : It contains TypeScript. It controls the dynamic data to display in components.

**Guards** are used for protecting the routes from unauthorised users to access it. Like “*Dashboard Page*” will be visible only when users login to the application, guards help in it.

**Services** are used for handling the state and any HTTP request. Creating different classes for it can be injected into any component and can be used.

**Models** are the just simple interfaces that are used as a data type in the application.

**Modules** contain multiple components and imports external libraries that are being used by the different components.

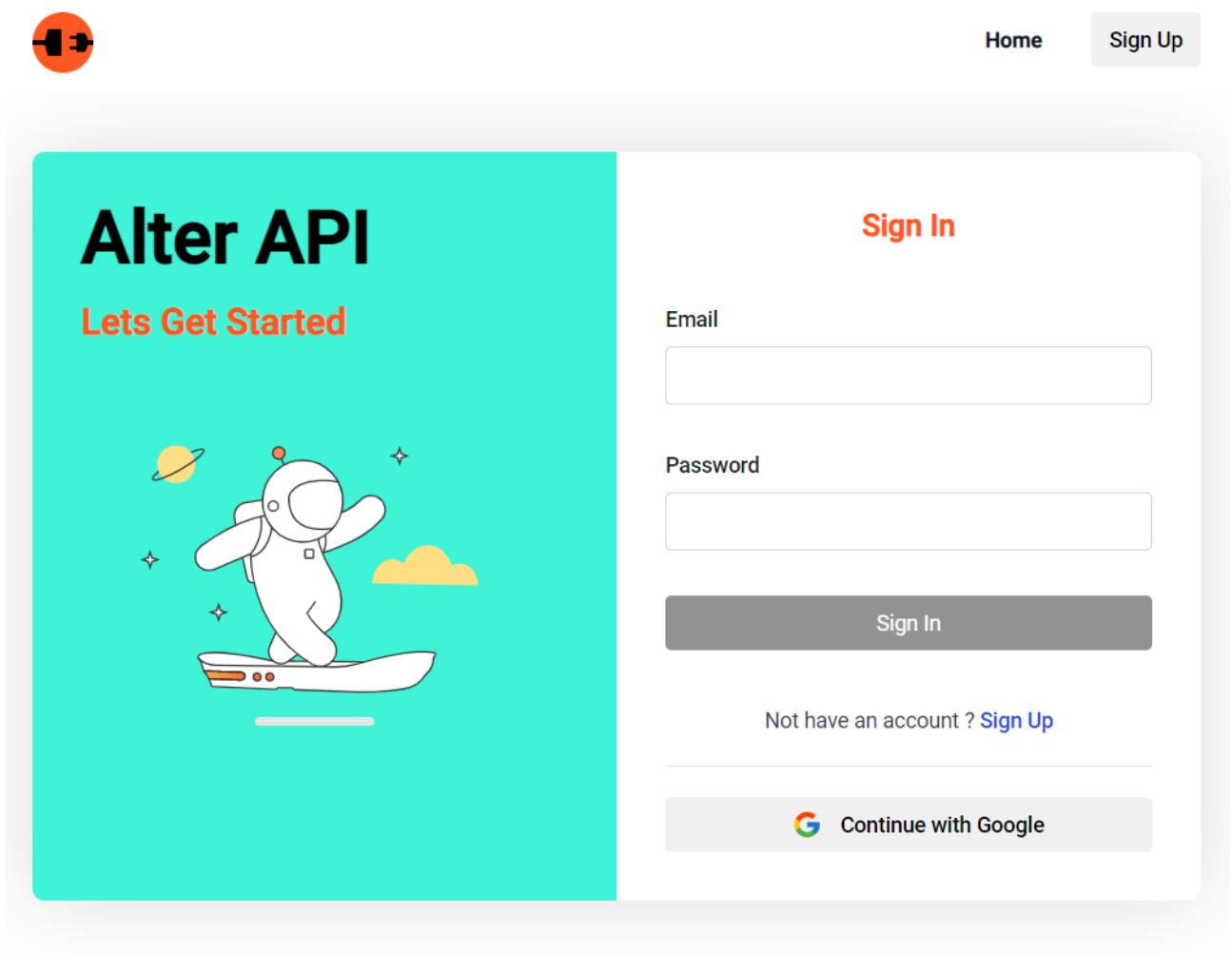
## 4.4 Form Design

Throughout the whole application there are different forms used for different functionalities.

Here are the list of form design in the application :

### 4.4.1 Login Form Design

Form used for logging the users into the application and can access the functionalities. It is a well validated form.



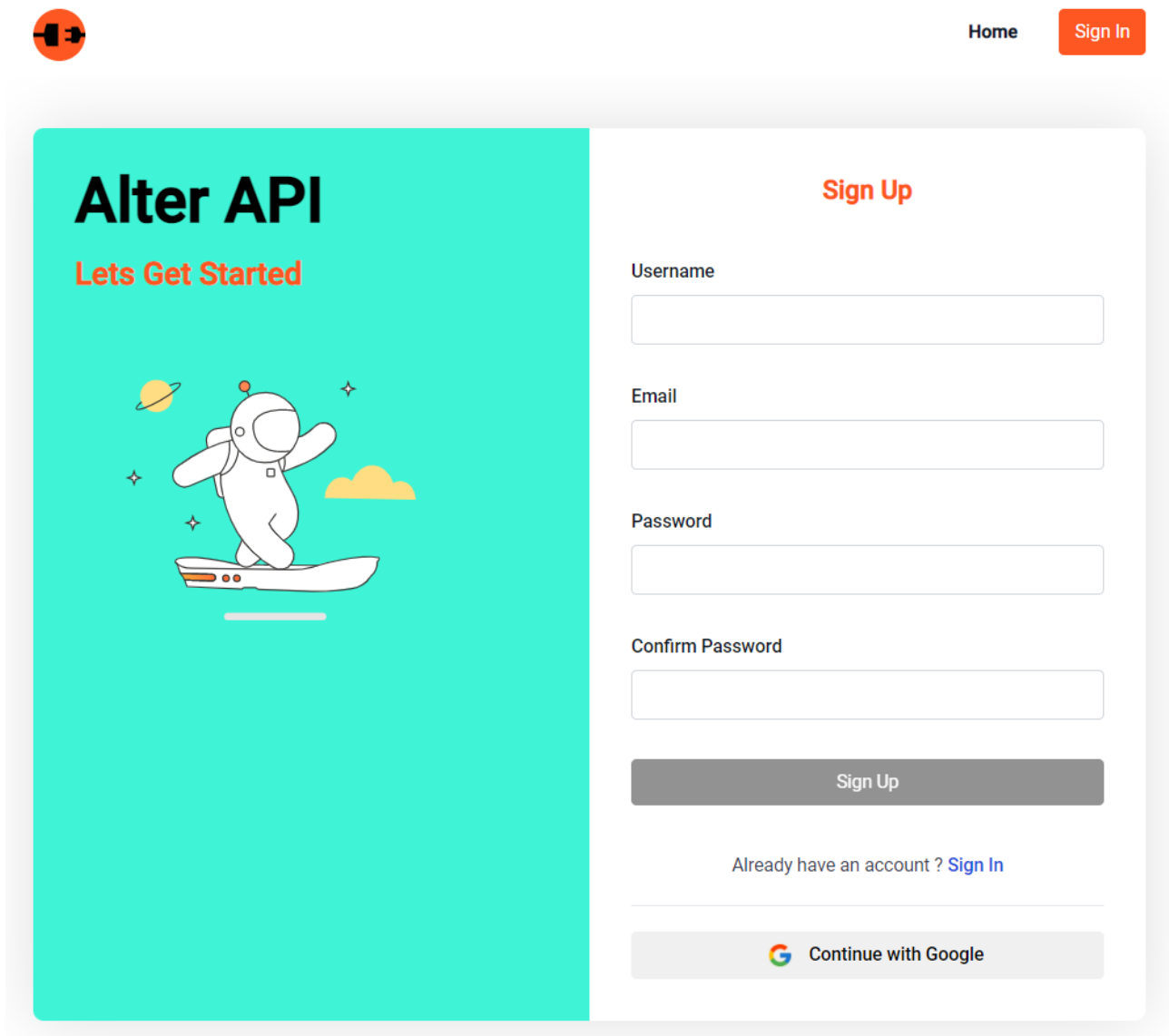
The image shows a login form for 'Alter API'. At the top left is a red circular logo with a white puzzle piece icon. To its right are links for 'Home' and 'Sign Up'. The main content area is split into two panels. The left panel has a teal background with the text 'Alter API' in large black font and 'Lets Get Started' in orange font below it. An illustration of an astronaut on a surfboard is centered in this panel. The right panel has a white background with the title 'Sign In' in orange. It contains two input fields labeled 'Email' and 'Password'. Below these is a grey 'Sign In' button. Further down is a link 'Not have an account ? Sign Up' in blue. At the bottom is a button with the Google logo and the text 'Continue with Google'.

Fig 4.6 Login Form design

#### 4.4.2 Register Form Design

This form is used for registering the users ( creating new users ) into the application, after entering the valid details, it checks for already existing users and displays errors.

After registering to the application, it will redirect users into the application no need.



The image shows a web application interface for registering a new user. At the top left is a red circular logo with a white arrow pointing right. At the top right are two links: "Home" and "Sign In", with "Sign In" highlighted in an orange button. The main content area is split into two columns. The left column has a teal background and contains the text "Alter API" in large black font, "Lets Get Started" in orange font, and a cartoon illustration of an astronaut surfing on a wave with a planet and stars in the background. The right column has a white background and is titled "Sign Up" in orange. It contains four input fields labeled "Username", "Email", "Password", and "Confirm Password". Below these fields is a grey "Sign Up" button. Underneath the button is a link: "Already have an account ? [Sign In](#)". At the bottom of the right column is a button with the Google logo and the text "Continue with Google".

Fig 4.7 Register Form design

### 4.4.3 Create Workspace Form Design

This form is used for adding the workspace under users account and can create different workspace for different projects.

It has functionality to invite the users just by adding the emails into the form and can choose which type of workspace you want.

---

## Create New Workspace

Name

dev-project-example

Summary

This is just an example workspace .

Type of Workspace

☐ Personal ☒ Team

Invite Members *( use SPACE, COMMA, ENTER for adding the email )*

example@xyz.com

deepak@gamil.com



ashish.98@gamil.com

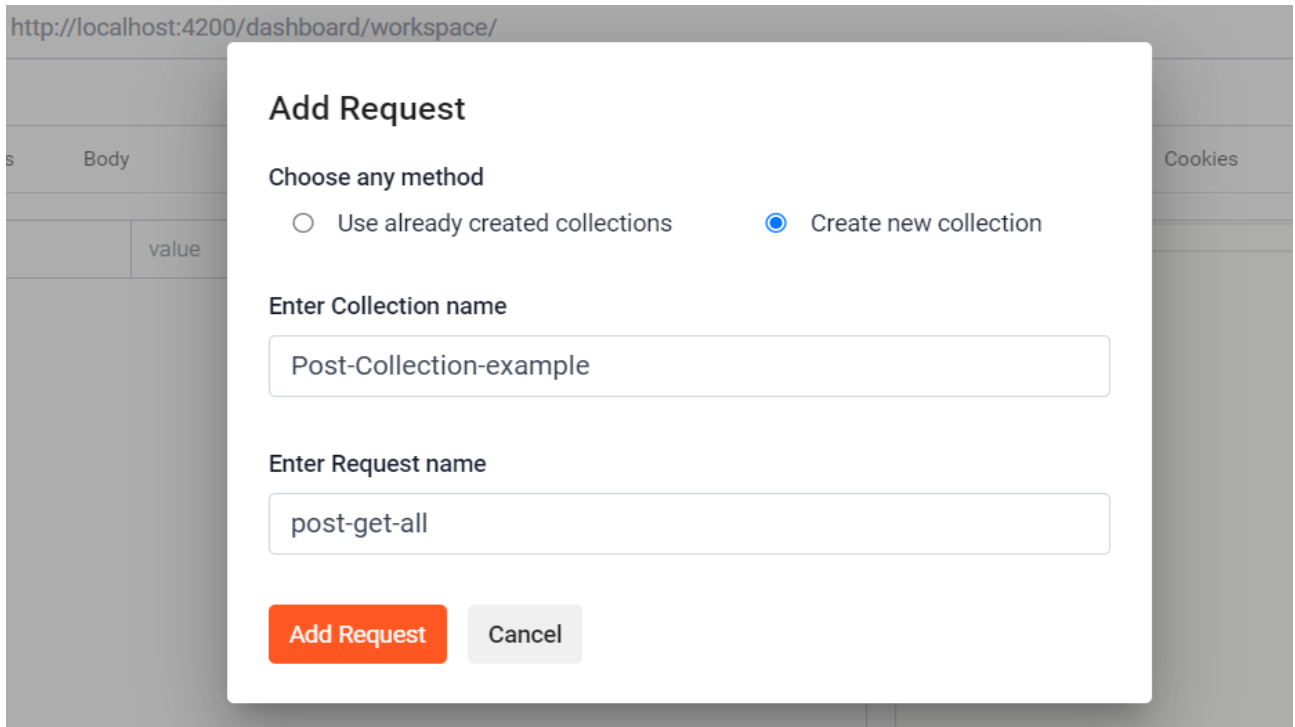


Create Workspace

Fig 4.8 Create Workspace Form design

#### 4.4.4 Creating new Collection and Request Form Design

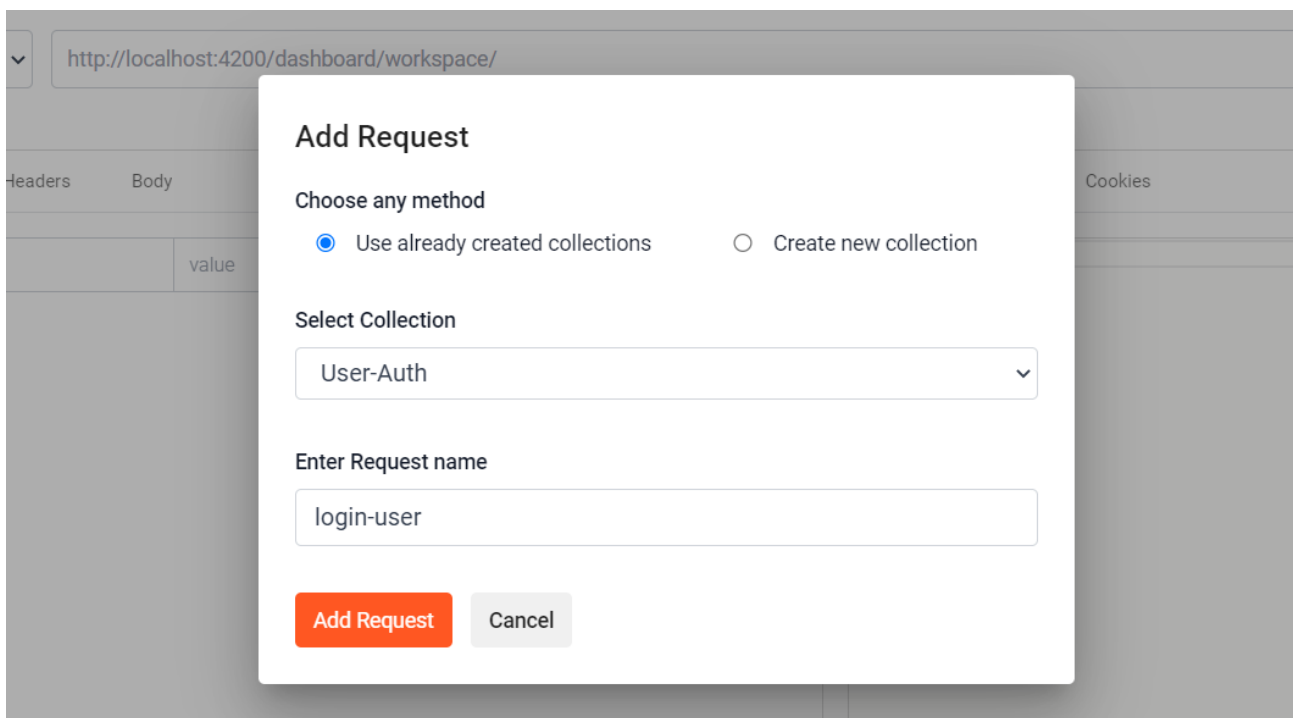
This form is a dialog form, used to create a new collection into any workspace users want. It can be used for adding a new Request tab into any collection for simulating the API .



The screenshot shows a web browser window with the URL `http://localhost:4200/dashboard/workspace/`. A modal dialog titled "Add Request" is open. It contains the following elements:

- Choose any method:** Two radio buttons. The first is "Use already created collections" (unselected). The second is "Create new collection" (selected with a blue dot).
- Enter Collection name:** A text input field containing the text "Post-Collection-example".
- Enter Request name:** A text input field containing the text "post-get-all".
- Buttons:** An orange "Add Request" button and a grey "Cancel" button.

Fig 4.9 Add Request Form (a)



The screenshot shows the same web browser window. The "Add Request" modal dialog is open, but with different selections:

- Choose any method:** The "Use already created collections" radio button is now selected (blue dot), and "Create new collection" is unselected.
- Select Collection:** A dropdown menu is present, showing "User-Auth" with a downward arrow.
- Enter Request name:** A text input field containing the text "login-user".
- Buttons:** The same orange "Add Request" and grey "Cancel" buttons are at the bottom.

Fig 4.10 Add Request Form (b)

#### 4.4.5 Inviting Users Form Design

This form is used to invite the users into any workspace as a user wants. Users just need to add the email of that user and select the workspace in which the incoming user will be invited.

Users can add multiple emails into the form.

This form can be accessed from any part of the application.

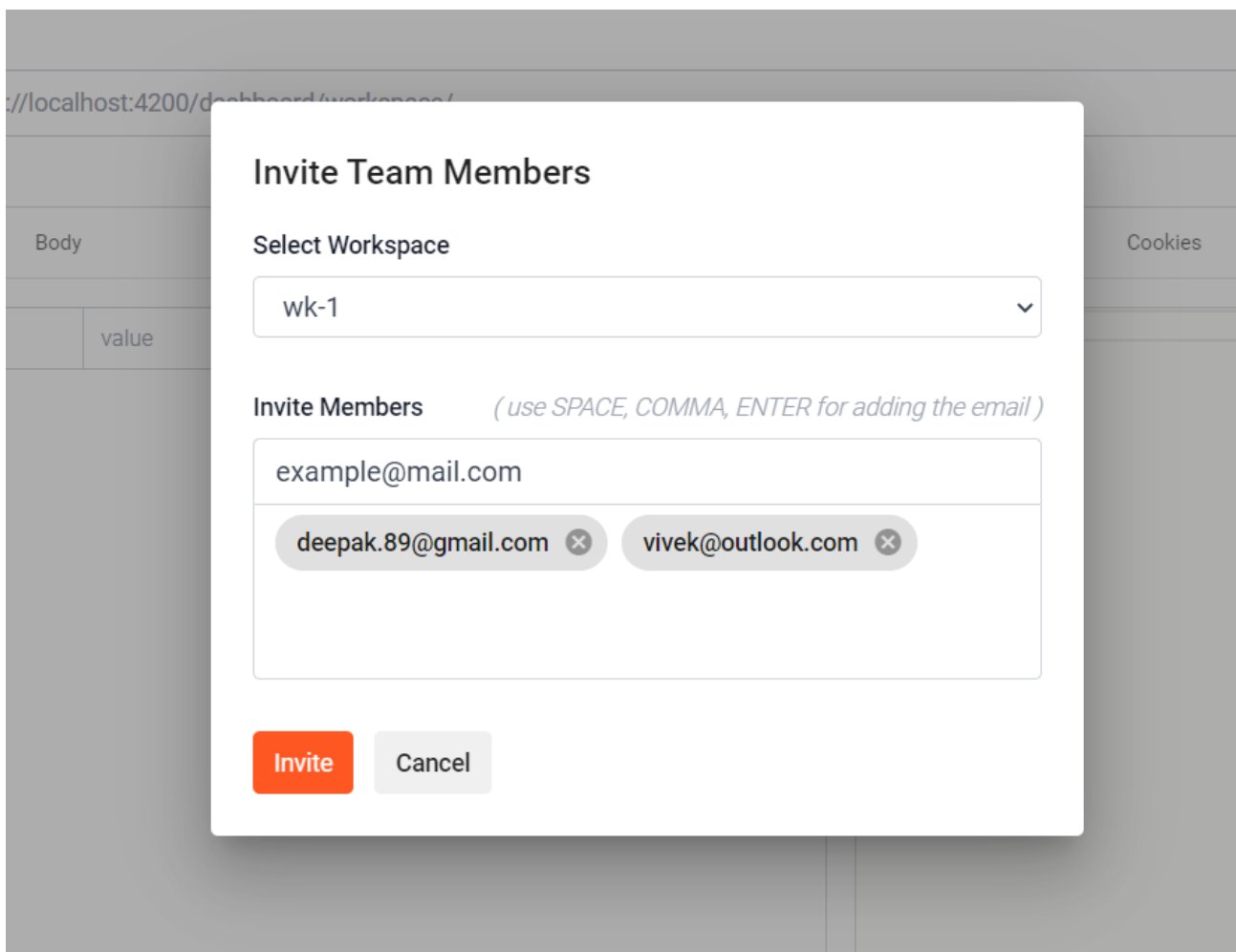
The image shows a modal dialog box titled "Invite Team Members" overlaid on a blurred background of a web application. The dialog has a white background and a subtle shadow. At the top, the title "Invite Team Members" is in a bold, dark font. Below the title is a section labeled "Select Workspace" with a dropdown menu currently showing "wk-1". Underneath this is a section labeled "Invite Members" with a hint in parentheses: "( use SPACE, COMMA, ENTER for adding the email )". This section contains a text input field with "example@mail.com" and two email tags below it: "deepak.89@gmail.com" and "vivek@outlook.com", each with a small 'x' icon to remove it. At the bottom of the dialog are two buttons: an orange "Invite" button and a light gray "Cancel" button.

Fig 4.11 Invite User Form design

## 4.5 User Interface Design

There are many interfaces for different purposes. Here is the list of it :

### 4.5.1 Landing Page

This is the landing page of the application, it contains all the information about What is the project about? How does the project work? What are the features of the application?



Fig 4.12 Landing Page (About)

### Features

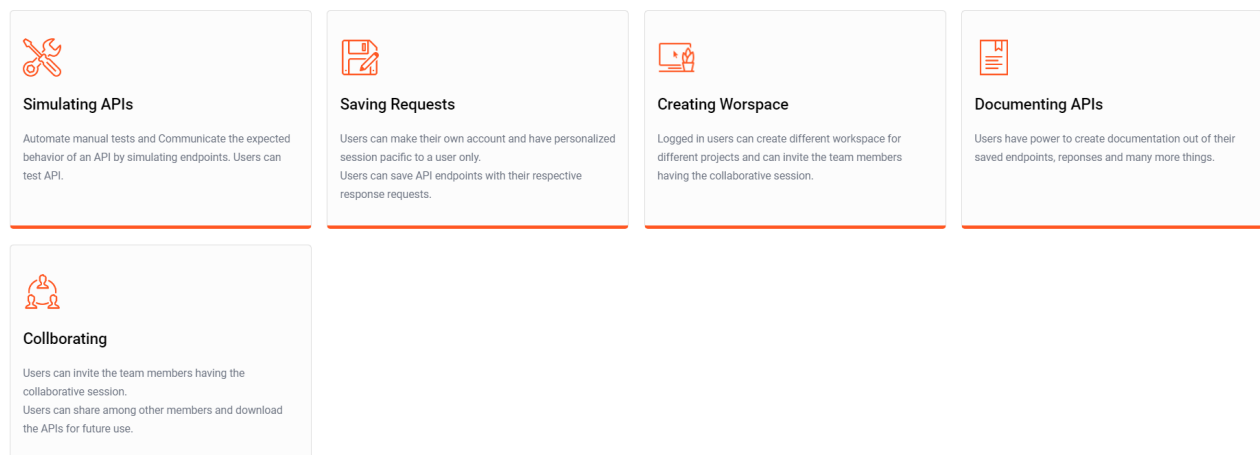


Fig 4.13 Landing Page (Feature)



### 4.5.2 Dashboard of a User.

This is a Dashboard of a user that contains the list of workspaces the user owns or working in.

The list of workspaces are expandable and can see the information about it.

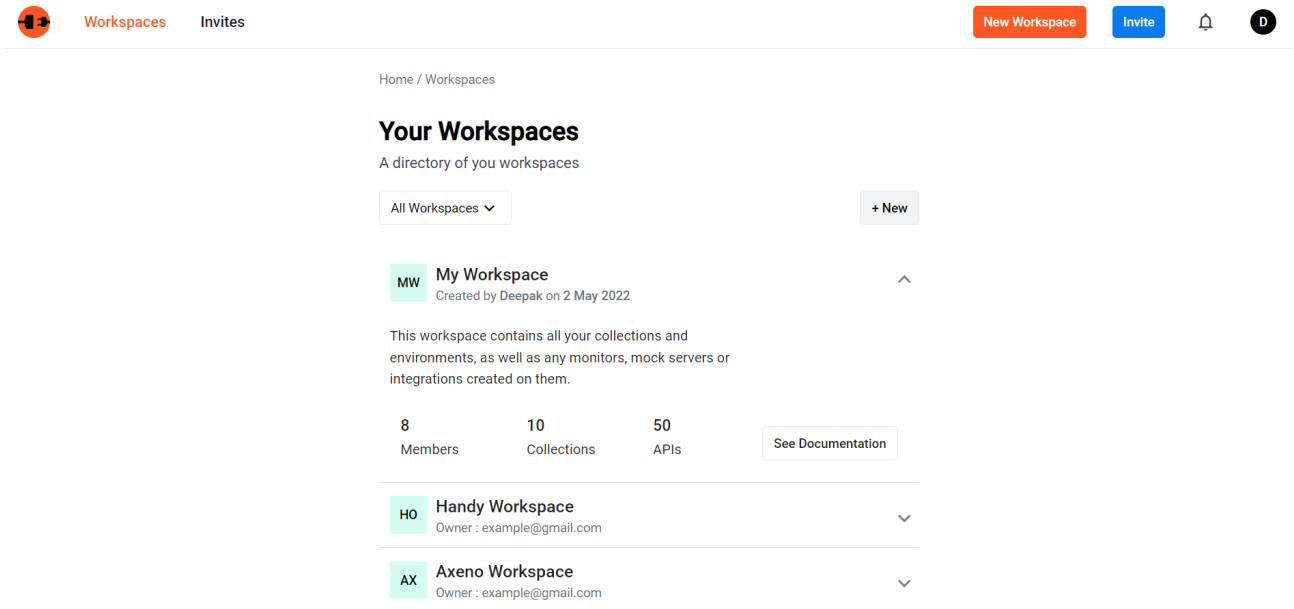


Fig 4.14 Workspace List

### 4.5.2 Invites Page of a Workspace.

Here all the invited members are listed into two categories : **Pending** and **Accepted** Invites.

Users can switch between different workspaces and see all the invited users. Admin can monitor the incoming members and invite more if needed for the project.

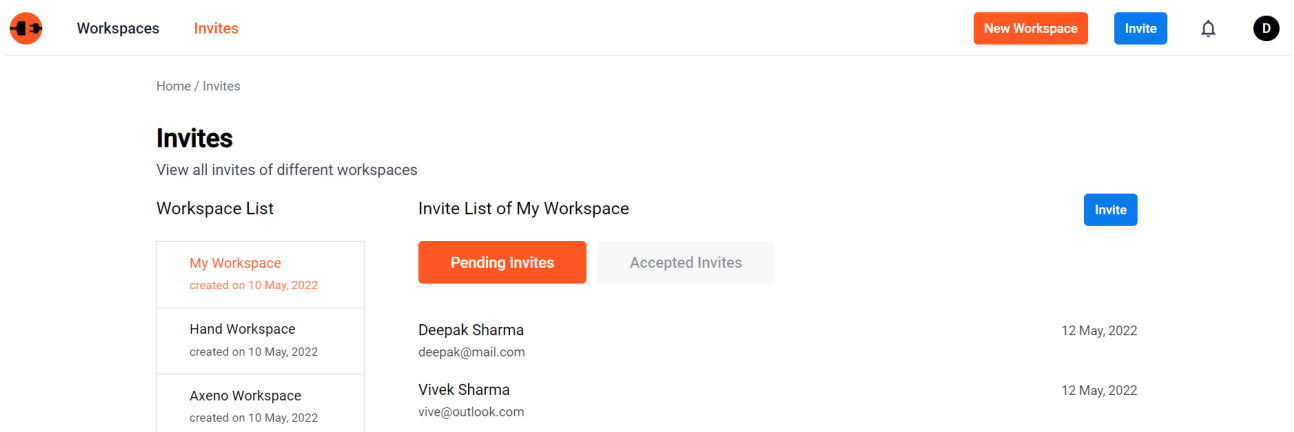


Fig 4.15 Invitation Page

### 4.5.3 Request Page of a Workspace.

This is the most important section of the application. All the magic happens here. This page is has many functionalities :

1. This is used to create new collections and requests to simulate them.
2. Users can create and open new tabs which have their own sub sections.
3. There is a side navigation, which contains a tree of collections and requests.
4. Each tab contains a URL input , Method choose, Request and Response area is divided into two equal parts which are customisable.
5. Request and Response sections have their own tabs for body, headers, query params.
6. After simulating the API, users can see responses on the same page only.

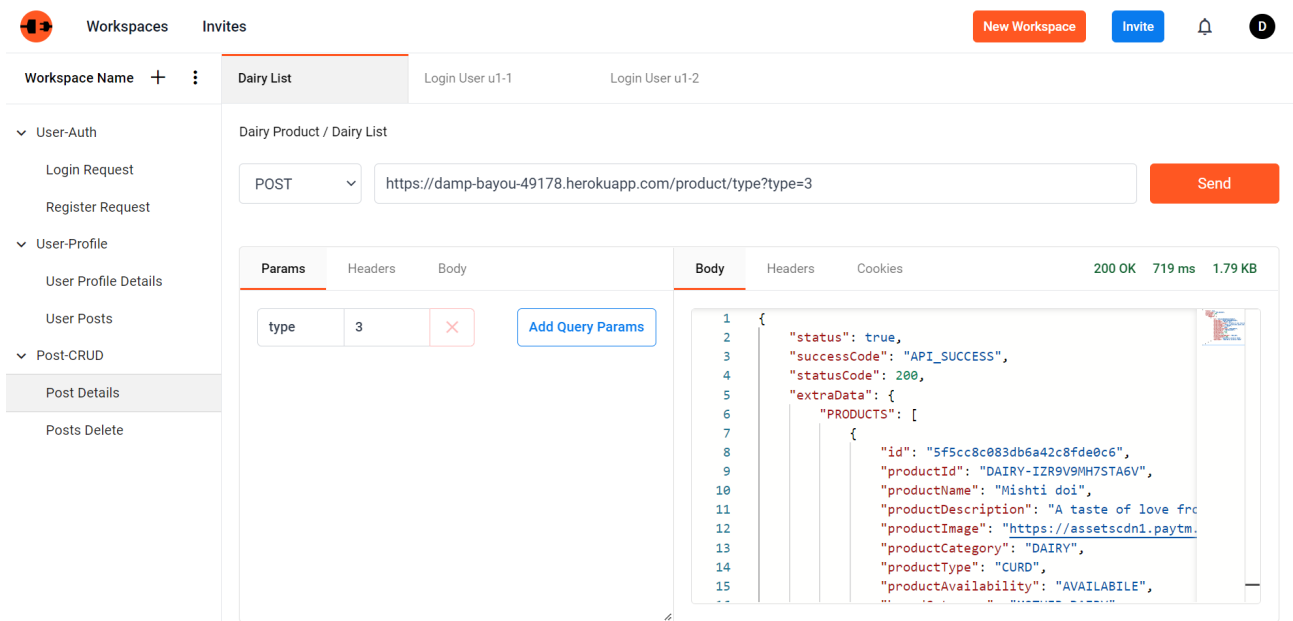


Fig 4.16 Request Tab Page (a)

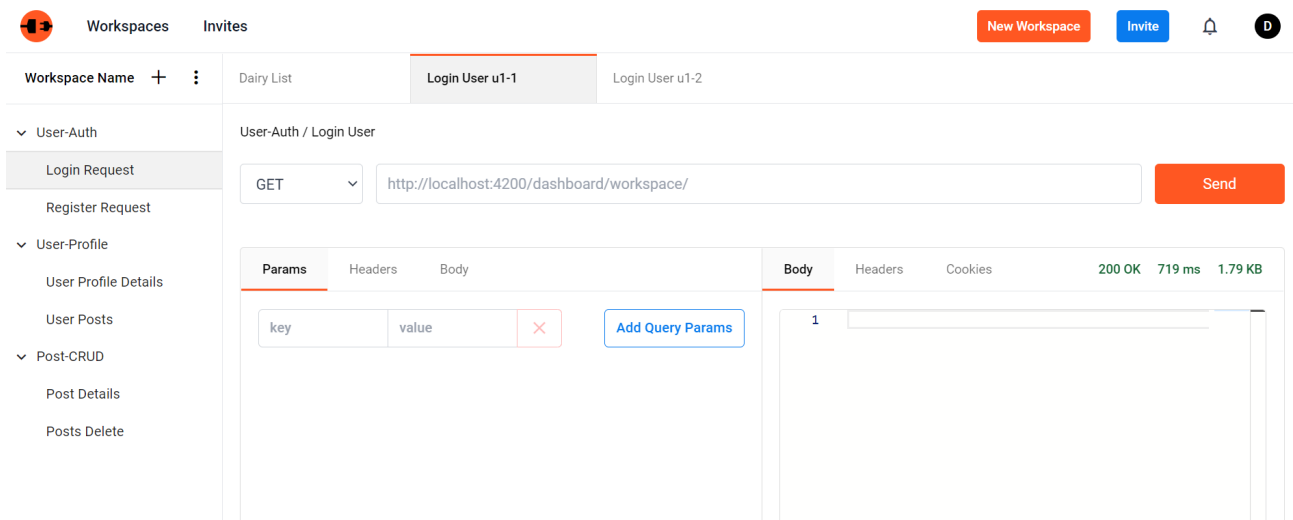


Fig 4.17 Request tab Page (b)

#### 4.5.4 Documentation of a Workspace.

This page is a documentation page for a particular workspace and it contains all the information about a workspace. It displays the information like header, params, body , response and many other things at one place with an example also.

This page is used by all the members of a workspace , and can monitor the APIs that are being simulated and tested.

The screenshot shows a workspace documentation page. At the top, there's a header with 'Workspaces' and 'Invites' tabs, and buttons for 'New Workspace' and 'Invite'. Below the header, the 'Workspace name' is displayed with a subtitle 'summary of the workspace'. The main content area is titled 'Product Collection' and shows a 'POST All Product List' request. The URL is 'https://damp-bayou-49178.herokuapp.com/product/type?type=3'. The 'Query Params' section shows 'type' with a value of '3'. The 'Response' section is expanded, showing the 'Body' tab with a JSON response. The response is a 200 OK status with a 719 ms response time and 1.79 KB of data. The JSON body contains metadata about the API and a 'scripts' section with commands for running the application.

Workspace name  
summary of the workspace

Product Collection

POST All Product List

https://damp-bayou-49178.herokuapp.com/product/type?type=3

Query Params

type 3

Response

Body Headers Cookies 200 OK 719 ms 1.79 KB

```
{
  "name": "alter-api",
  "version": "0.0.0",
  "homepage": "https://deeps8.github.io/AlterAPI-Tester",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "watch": "ng build --watch --configuration development",
    "test": "ng test",
    "deploy": "gh-pages -d dist\\Alter-API"
  },
  "private": true,
}
```

Fig 4.18 Documentation Page (a)

The screenshot shows a workspace documentation page, similar to Fig 4.18. The 'Response' section is expanded, showing the 'Headers' tab. The response is a 200 OK status with a 719 ms response time and 1.79 KB of data. The headers section shows a table with 'Key' and 'Value' columns, containing 'Key-1' and 'value-1'.

Workspace name  
summary of the workspace

Product Collection

POST All Product List

https://damp-bayou-49178.herokuapp.com/product/type?type=3

Query Params

type 3

Response

Body Headers Cookies 200 OK 719 ms 1.79 KB

Key	Value
Key-1	value-1

Fig 4.19 Documentation Page (b)

# **Chapter 5**

## **Project Management and Scheduling**

### **5.1 Project Planning and Scheduling**

#### **5.1.1 Gantt Chart**

### **5.2 Risk Management**

Project management involves the planning, monitoring and control of the people, process and events that occur as software evolves from a preliminary concept to an operational implementation.

Project managers plan, monitor and control the work of a team of software engineers.

Effective software project management focuses on the four P's: people, product, process, and project.

### **Function Points**

The function point (FP) metric can be used effectively as a means for measuring the functionality delivered by a system. Using historical data, the FP metric can then be used to:

- (1) Estimate the cost or effort required to design, code, and test the software.
- (2) Predict the number of errors that will be encountered during testing.
- (3) Forecast the number of components and the number of projected source lines in the implemented system.

Function points are derived using an empirical relationship based on countable (direct) measures of software's information domain and qualitative assessments of software complexity.

### **Information domain values are defined in the following manner:**

*Number of external inputs (EIs).* Each external input originates from a user or is transmitted from another application and provides distinct application-oriented data or control information. Inputs are often used to update internal logical files (ILFs). Inputs should be distinguished from inquiries, which are counted separately.

*Number of external outputs (EOs).* Each external output is derived data within the application that provides information to the user. In this context external output refers to reports, screens, error messages, etc. Individual data items within a report are not counted separately.

*Number of external inquiries (EQs).* An external inquiry is defined as an online input that results in the generation of some immediate software response in the form of an online output (often retrieved from an ILF).

*Number of internal logical files (ILFs).* Each internal logical file is a logical grouping of data that resides within the application's boundary and is maintained via external inputs.

*Number of external interface files (EIFs).* Each external interface file is a logical grouping of data that resides external to the application but provides information that may be of use to the application.

Once these data have been collected, a complexity value is associated with each count. Organisations that use function point methods develop criteria for determining whether a particular entry is simple, average, or complex. Nonetheless, the determination of complexity is somewhat subjective.

To compute function points (FP), the following relationship is used:

$$FP = \text{count total} * [0.65 + 0.01 * (Fi)]$$

Where,

- Count total = Sum of the product obtained from multiplying counts of each information domain value by a weighting factor depending on their complexity level.
- $F_i$  ( $i = 1$  to  $14$ ) = "Complexity Adjustment Values" based on responses to a set of 14 questions; each answered using a scale ranging from 0 to 5:

0 = No influence

1 = Simple

2 = Moderate

3 = Average

4 = Significant

5 = Essential

$$FP = \text{count total} * [0.65 + 0.01 * (Fi)]$$

$$FP = (68) * [0.65 + 0.01 * (48)]$$

**Function Point = 76.84**

## 5.1 Gantt Chart

When creating a software project schedule, we begin with a set of tasks (the work breakdown structure). If automated tools are used, the work breakdown is input as a task network or task outline. Effort, duration, and start date are then input for each task. In addition, tasks may be assigned to specific individuals.

As a consequence of this input, a time-line chart, also called a Gantt chart, is generated. A time-line chart can be developed for the entire project. Alternatively, separate charts can be developed for each project function or for each individual working on the project.



Fig 5.1 Gantt Chart

## 5.2 Risk Management

Risk analysis and management are actions that help a software team to understand and manage uncertainty. Many problems can affect a software project. Risk is considered as a probability that some adverse circumstances will occur. Software is a difficult undertaking and is also a key to good software project management. Risk analysis is done by everyone who is involved in the process of the project.

### Risk Identification

Risk identification refers to the process of identifying dangerous or hazardous situations and trying to characterise it. It is a procedure to analyse, review and anticipate possible risks.

There are many types of risk identification which helps in the project:

S.No.	Risk	Description
1	Technology Risk	If the platform requirement is not fulfilled, then may be the system fails to execute.
2	User Risk	Users cannot use this System if they are unknown and don't have any idea regarding the same.
3	Organisational Risk	Poor communication always leads to the problems between the workers within the organisation
4	Estimation Risk	The cost estimated for developing the system may exceed the expectations resulting in budget imbalance
5	Used Tools Risk	The HMS will operate only on Visual Studio 2008 and hence if the platform is not present in other organisations, this system fails to operate.



## Risk Analysis

Risk analysis should be performed as part of the risk management process for each project. The probability and the seriousness of the risk in the “Collaborative API Software : Alter-API” might be assessed as low or high.

<b>Risk</b>	<b>Probability (High / Low)</b>	<b>Effects</b>
Technology risks	High	System crash/ failure.
User risk	Low	Lack of user knowledge to use the system.
Organisational risk	Low	Mental and health problems & communication failure among workers in the organisation.
Estimation risk	High	Budget imbalance.
User Tools risk	Low	System failure in execution

## Risk Prioritisation

Risk Prioritisation is done according to seriousness of the risk and once the risk has been analysed and ranked, a judgement must be made that which are the most important risks that must be considered during the project development. The risks are prioritised according to the value of the Risk Exposure.

Risk Marking Scheme:      Low: 0-3      |      Medium: 4-6 |      High: 7-10

### **Risk Exposures**

<b>S.No.</b>	<b>Risk</b>	<b>Risk Exposure</b>
1	Software corruption	7
2	Inexperienced instructor	6.5
3	Design & implementation risks	4.75
4	Application and complexity	2.8
5	High workload risk	2.15
6	High project size	1.75
7	Maintenance problems	1.8

# **Chapter 6**

## **Testing and Deployment**

### **6.1 Testing**

### **6.2 Deployment**

## 6.1 Testing

Testing is to check, verify and evaluate the product that comes up after going through the demanding processes. Since the designed system has both Online/Offline features, the type of testing that should be implemented for the system are Alpha testing, Beta testing, Unit testing, Integration tests and System testing.

### 6.1.1 Black Box testing

*BLACK BOX TESTING*, also known as Behavioural Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.

This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see. This method attempts to find errors in the following categories:

- Incorrect or missing functions
- Interface errors
- Errors in data structures or external database access
- Behaviour or performance errors
- Initialization and termination errors

### 6.1.2 White box testing

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing). In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases.

White-box testing can be applied at the unit, integration and system levels of the software testing process. Although traditional testers tended to think of white-box testing as being done at the unit level, it is used for integration and system testing more frequently today.

It can test paths within a unit, paths between units during integration, and between subsystems during a system-level test. Though this method of test design can uncover many errors or problems, it has the potential to miss unimplemented parts of the specification or missing requirements.

### 6.1.3 Basis Path Testing

Basis path testing, a structured testing or white box testing technique used for designing test cases intended to examine all possible paths of execution at least once. Creating and executing tests for all possible paths results in 100% statement coverage and 100% branch coverage.

#### Login Page Testing :

Test Case Name		Login Module Testing					
Purpose of Test					Check the correctness & Integrity of Login Module.		
Test Attribute					By default the cursor will be in the User Name text field. After entering the User Name, the cursor should switch to the password field by using Tab Key or by clicking on the password textbox. Sign in option is there to log into the system after fulfilling mandatory details.		
Test Focus	Function	Feature	Process	Interface	Validation	Verification	
		Whether the complete Login module is working properly or not.			Proper validation for User ID, Password and other fields are given or not.	At last the complete Login module is verified.	
Test Type				Unit Testing			
Test Process	Initiation	Starting Condition	Input Specific ation	Outputs Expected	Assumptions Made		

User starts the system from the computer.		Screen shows login module i.e. admin login form		User ID, password are entered. Submit button clicked.		Successful Login. Failed login due to incorrect field entry. Failed login due to fields not filled		Passwords has minimum 6 characters.			
Test Results		Criteria		Expected Result		Actual Result		Error Description		State	
User ID		As per database entry		As per database entry		-		Error Free State			
Password		As per database entry & both the attributes should match.				In some cases, both the attributes don't match.					
Action						In case of any error, user has the option to click on the forgot password and retrieve the password by filling in the mandatory details.					

### Source Code for testing ( Login Page ) :

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { AuthComponent } from './auth.component';

describe('AuthComponent', () => {
  let component: LoginComponent;

  beforeEach(async() => {
    component = new LoginComponent(routerSpy, new FormBuilder(), loginServiceSpy);
  });

  function updateForm(userEmail, userPassword) {
    component.loginForm.controls['username'].setValue(userEmail);
    component.loginForm.controls['password'].setValue(userPassword);
  }
});
```

```

}

it('Component successfully created', () => {
  expect(component).toBeTruthy();
});

it('component initial state', () => {
  expect(component.submitted).toBeFalsy();
  expect(component.loginForm).toBeDefined();
  expect(component.loginForm.invalid).toBeTruthy();
  expect(component.authError).toBeFalsy();
  expect(component.authErrorMsg).toBeUndefined();
});

it('submitted should be true when onSubmit()', () => {
  component.onSubmit(blankUser);
  expect(component.submitted).toBeTruthy();
  expect(component.authError).toBeFalsy();
});

it('form value should update from when u change the input', () => {
  updateForm(validUser.username, validUser.password);
  expect(component.loginForm.value).toEqual(validUser);
});

it('Form invalid should be true when form is invalid', () => {
  updateForm(blankUser.username, blankUser.password);
  expect(component.loginForm.invalid).toBeTruthy();
});
});

```

## Register Page Testing :

Test Case Name		Register Module Testing				
Purpose of Test				Check the correctness & Integrity of Register Module.		
Test Attribute				By default the cursor will be in the User Email text field. Steps for registering : 1. Email 2. Username 3. Password 4. Confirm Password		
Test Focus	Function	Feature	Process	Interface	Validation	Verification
		Whether the complete Register module is working properly or not.			Proper validation for User ID, Password and other fields are given or not.	At last the complete Register module is verified.
Test Type				Unit Testing		
Test Process	Initiation	Starting Condition		Input Specific ation	Outputs Expected	Assumptions Made
User starts the system from the computer.	Screen shows Register Module	User Email, User Name, password are entered. Submit button clicked.		Successful Register. Failed register due to incorrect field entry. Failed register due to fields not filled	Passwords has minimum of 6 characters.	



Test Results	Criteria	Expected Result	Actual Result	Error Description	State
User Name	As per database entry	As per database entry	-		Error Free State
Email	As per database entry	As per database entry	-		Error Free State
Password	As per database entry & both the attributes should match.		In some cases, both the attributes don't match.		
Action			In case of any error, the user has the option to re-enter the details and Register / Submit form again.		

### Create Workspace Form Page Testing :

Test Case Name		Workspace Module Testing					
Purpose of Test					Check the correctness & Integrity of Workspace Module with its forms and display.		
Test Attribute					By default the user will be displayed the list of workspaces. - Users have freedom to navigate anywhere. - Creating a new Workspace with the “New Workspace ” button.		
Test Focus	Function	Feature	Process	Interface	Validation	Verification	
Workspace form  UI of the workspace	Creating a new workspace and validating all the fields	Whether the Workspace module is working properly or not.		Total of 6 Process	Proper validation for Workspace form and display all things.	At last the complete Workspace module is verified.	

Test Type				Unit Testing			
Test Process	Initiation		Starting Condition		Input Specific ation	Outputs Expected	Assumptions Made
After Logging / Registering Users get redirected to the Workspace or a Dashboard.		Screen shows Workspace Module		Users interact with the workspace list and open the accordions. User navigates to the “Create Workspace“ form.		Successful List open.	There is list of workspaces in the Dashboard
Create Workspace Form		Click on the button on the header.		User Enter the fields as provided by the application.		On Success a new Workspace will be created. On failure , it shows the error on top of the form.	User enters the valid details as per the form.
Test Results	Criteria		Expected Result		Actual Result	Error Description	
Workspace Name		Any new name.		As per database entry		Required, Short name	
Summary		Any string		As per database entry		Required, Short summary	
Type of the Workspace		As per database entry & both the attributes should match.			Workspace type not selected		
Emails for inviting the members.		As per database, no new user emails.			No email provided. Member not available.		
Action					In case of any error, the user has the option to re-enter the details and Submit form again.		

## 6.2 Deployment

This application is (will be) deployed on Firebase only. Firebase provides **Firebase Hosting**, with the help of it we can deploy the application.

**Firebase Hosting** is built for the modern web developer. Websites and apps are more powerful than ever with the rise of front-end JavaScript frameworks like Angular and static generator tools like Jekyll. Whether you are deploying a simple app landing page or a complex Progressive Web App (PWA), Hosting gives you the infrastructure, features, and tooling tailored to deploying and managing websites and apps.

As the project is integrated with **Angular-Fire** a module for connecting to Firebase and its other services. We can just use some commands for deploying the application.

### Steps for deploying the application :

1. Firstly configure the application with firebase tools and install the required modules.
2. Commands for deployment :
  - a. *Firebase deploy*
3. Enter the asked details in the command line.
4. And we are all set , Application is deployed on the **Firebase**.

# **Chapter 7**

## **Result and Conclusion**

7.1 Conclusion

7.2 Limitation

7.3 Future Scope of Project

7.4 Reference

## 7.1 Conclusion

This project was designed to make developers' lives easier and efficient by providing them with software for testing, building and documenting APIs which are the backbone of any application.

Providing the collaborative feature to the app took it to the next level .It is not just limited to individual users, but to a whole team interacting with each other and developing bug free APIs.

It also helps in understanding how API works , with that information developers can integrate into the front-end of the app.

The following conclusions can be deduced from the development of the project:

- Automation of the entire system improves the efficiency
- It provides a friendly graphical user interface which proves to be better when compared to the existing system.
- It gives appropriate access to the authorised users depending on their permissions.
- It effectively overcomes the delay in communications.
- Updating information becomes so easy.

## **7.2 Limitation**

- People who are not Techno-friendly cannot make the best use of this system.
- Even after automation of the system, the scope for human errors still persists as the information entered by customer/executive may be flawed.
- The server has not been created to process a colossal amount of user requests at a time.
- Much attention has not been paid to the Loading time of the Web Application.

## **7.3 Future Scope of the Project**

- Improving the request simulation , with new features and better performance.
- Providing users to add a new test server for handling the request and many other types of options along with it.
- Adding a new functionality to the application , that users can simulate requests on different network ranges and can have better test cases.
- Adding new Option for both request and response sections.
- Providing a version control system for APIs.

## 7.4 Reference

1. <https://www.ibm.com/cloud/learn/api>
2. <https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces>
3. <https://aws.amazon.com/what-is/api/>
4. <https://learning.postman.com/docs/getting-started/introduction/>
5. <https://swagger.io/blog/api-documentation/what-is-api-documentation-and-why-it-matters/>
6. <https://www.altexsoft.com/blog/api-documentation/>
7. <https://www.bacancytechnology.com/blog/deploy-angular-12-application-using-firebase-hosting>