RAJALAKSHMI ENGINEERING COLLEGE

An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ACADEMIC YEAR 2024-2025

EVEN SEMESTER

CS23432 - Software Construction
Lab Manual

Name:	Kamalesh S P
Year / Branch / Section:	II year – CSE – B
Register No:	2116230701138
Semester:	IV
Academic Year:	2024 – 2025
Team:	K-TECH

LAB PLAN

CS234342 - SOFTWARE CONSTRUCTION LAB

Ex No	Date	Topic	Page No	Sign
1	30.01.2025	Study of Azure DevOps	1	
2	06.02.2025	Writing Problem Statement	3	
3	13.02.2025	Agile Planning	4	
4	20.02.2025	Creating User stories	6	
5	27.02.2025	Designing Sequence Diagram	11	
6	06.03.2025	Designing Class Diagram	13	
7	13.03.2025	Designing Use Case Diagram	20	
8	20.03.2025	Designing Activity Diagram	23	
9	27.03.2025	Designing Architecture Diagram	26	
10	03.04.2025	Design User Interface	29	
11	05.05.2025	Implementation – Design a Web Page based on Scrum Methodology	31	

Ex. No. 1	**STUDY OF AZURE DEVOPS**

AIM:

To study how to create an agile project in Azure DevOps environment.

STUDY:

Azure DevOps is a cloud-based platform by Microsoft that provides tools for DevOps practices, including CI/CD pipelines, version control, agile planning, testing, and monitoring. It supports teams in automating software development and deployment.

1. Understanding Azure DevOps

Azure DevOps consists of five key services:

1.1 Azure Repos (Version Control)

Supports Git repositories and Team Foundation Version Control (TFVC).

Provides features like branching, pull requests, and code reviews.

1.2 Azure Pipelines (CI/CD)

Automates build, test, and deployment processes.

Supports multi-platform builds (Windows, Linux, macOS).

Works with Docker, Kubernetes, Terraform, and cloud providers (Azure, AWS, GCP).

1.3 Azure Boards (Agile Project Management)

Manages work using Kanban boards, Scrum boards, and dashboards.

Tracks user stories, tasks, bugs, sprints, and releases.

1.4 Azure Test Plans (Testing)

Provides manual, exploratory, and automated testing.

Supports test case management and tracking.

1.5 Azure Artifacts (Package Management)

Stores and manages NuGet, npm, Maven, and Python packages.

Enables versioning and secure access to dependencies.

Getting Started with Azure DevOps

Step 1: Create an Azure DevOps Account

Visit Azure DevOps.

Sign in with a Microsoft Account.

Create an Organization and a Project.

Step 2: Set Up a Repository (Azure Repos)

Navigate to Repos.

Choose Git or TFVC for version control.

Clone the repository and push your code.

Step 3: Configure a CI/CD Pipeline (Azure Pipelines)

Go to Pipelines → New Pipeline.

Select a source code repository (Azure Repos, GitHub, etc.)

Define the pipeline using YAML or the Classic Editor.

Run the pipeline to build and deploy the application.

Step 4: Manage Work with Azure Boards

Navigate to Boards.

Create work items, user stories, and tasks.

Organize sprints and track progress.

Step 5: Implement Testing (Azure Test Plans)

Go to Test Plans.

Create and run test cases

View test results and track bugs

Result:

The study was successfully completed.

Ex. No. 2	# WRITING PROBLEM STATEMENT

AIM:

To prepare a **problem statement** for your given project.

Problem Statement:

Messaging Compilation System – Counselling for Students

Many students struggle with academic pressure, mental health concerns, and career-related uncertainties. Due to limited availability, long wait times, and social stigma associated with traditional counseling methods, students often lack timely and confidential support. This project proposes a Messaging Compilation System that facilitates accessible digital counseling for students. The system allows students to communicate their concerns via messages, which are compiled, categorized, and forwarded to counselors or mental health professionals for timely feedback. This tool aims to bridge the gap between students and support systems by providing a secure, stigma-free, and convenient platform for emotional and academic guidance.

Result:

The problem statement was written successfully.

Ex. No. 3	**AGILE PLANNING**

Aim:

To prepare an Agile Plan.

THEORY

Agile planning is a part of the Agile methodology, which is a project management style with an incremental, iterative approach. Instead of using an in-depth plan from the start of the project—which is typically product-related—Agile leaves room for requirement changes throughout and relies on constant feedback from end users.

With Agile planning, a project is broken down into smaller, more manageable tasks with the ultimate goal of having a defined image of a project's vision. Agile planning involves looking at different aspects of a project's tasks and how they'll be achieved, for example:

Roadmaps to guide a product's release ad schedule

1. Sprints to work on one specific group of tasks at a time

2. A feedback plan to allow teams to stay flexible and easily adapt to change

User stories, or the tasks in a project, capture user requirements from the end user's perspective Essentially, with Agile planning, a team would decide on a set of user stories to action at any given time, using them as a guide to implement new features or functionalities in a tool. Looking at tasks as user stories is a helpful way to imagine how a customer may use a feature and helps teams prioritize work and focus on delivering value first.

Steps in Agile planning process

1. Define vision

2. Set clear expectations on goals

3. Define and break down the product roadmap

4. Create tasks based on user stories

5. Populate product backlog

6. Plan iterations and estimate effort

7. Conduct daily stand-ups

8. Monitor and adapt

Result:

Thus, the Agile plan was completed successfully.

Ex. No. 4	**CREATING USER STORIES**

Aim:

To create User Stories

THEORY

A user story is an informal, general explanation of a software feature written from the perspective of the end user. Its purpose is to articulate how a software feature will provide value to the customer.

User story template:

"As a [role], I [want to], [so that]."

Procedure:

1. Open your web browser and go to the Azure website:
   https://azure.microsoft.com/en-in.
   Sign in using your Microsoft account credentials. If you don't have an account, you'll need to create one.

2. If you don't have a Microsoft account, you can sign up for
   https://signup.live.com/?lic=1
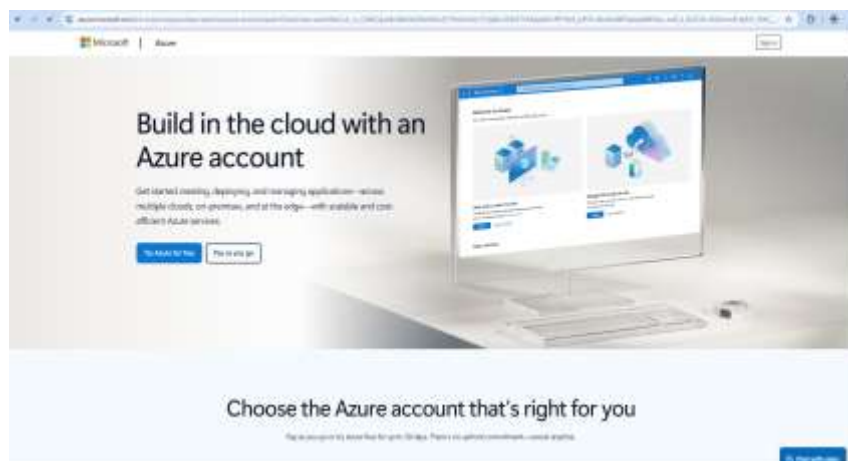
3. Azure Home Page

4. Open DevOps environment in the Azure platform by typing Azure DevOps Organizations in the search bar.
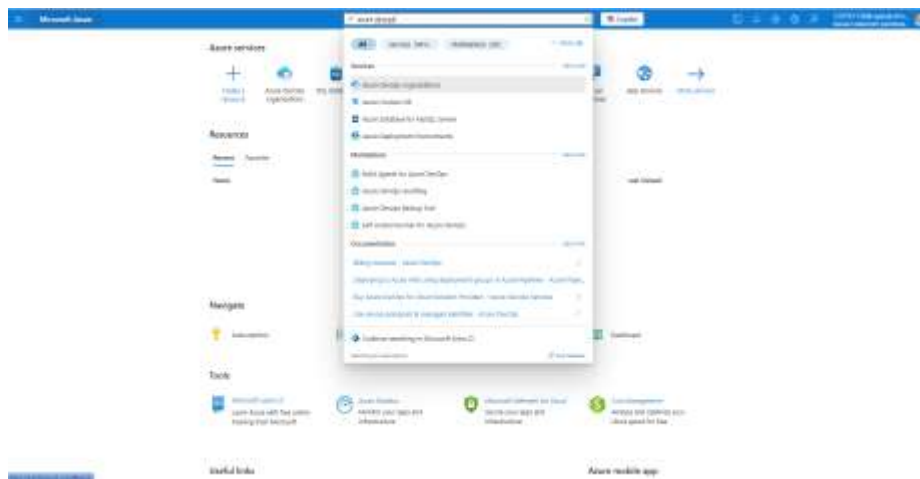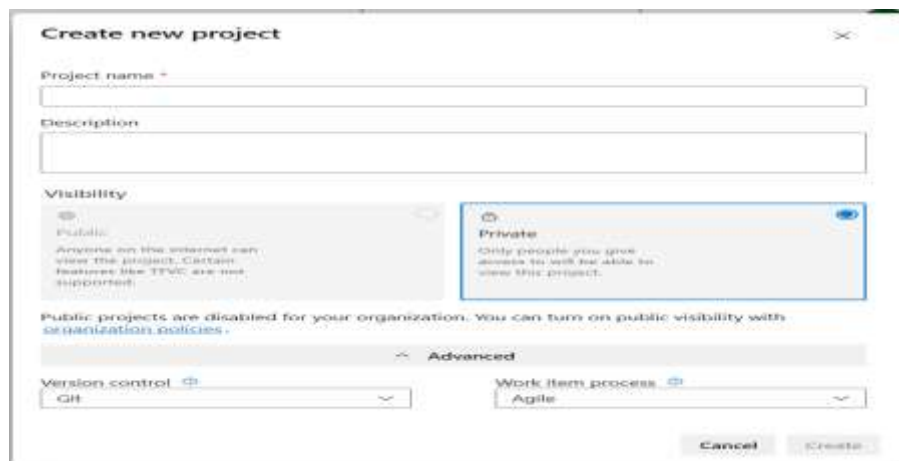
5. Click on the My Azure DevOps Organization link and create an organization and you should be taken to the Azure DevOps Organization Home page

6. Create the First Project in Your Organization

        After the organization is set up, you'll need to create your first project. This is where you'll begin to manage code, pipelines, work items, and more.

      i.   On the organization's Home page, click on the New Project button.

      ii.   Enter the project name, description, and visibility options:

          ○ Name: Choose a name for the project (e.g., LMS).

          ○ Description: Optionally, add a description to provide more context about the project.

          ○ Visibility: Choose whether you want the project to be Private (accessible only to those invited) or Public (accessible to anyone).

      iii.   Once you've filled out the details, click Create to set up your first project.

7. Once logged in, ensure you are in the correct organization. If you're part of multiple organizations, you can switch between them from the top left corner (next to your user profile). Click on the Organization name, and you should be taken to the Azure DevOps Organization Home page.
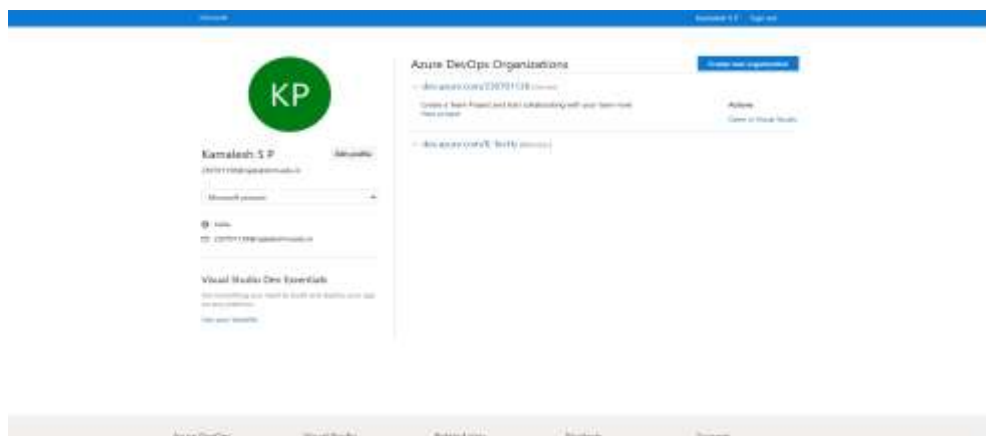
8. Project Dashboard
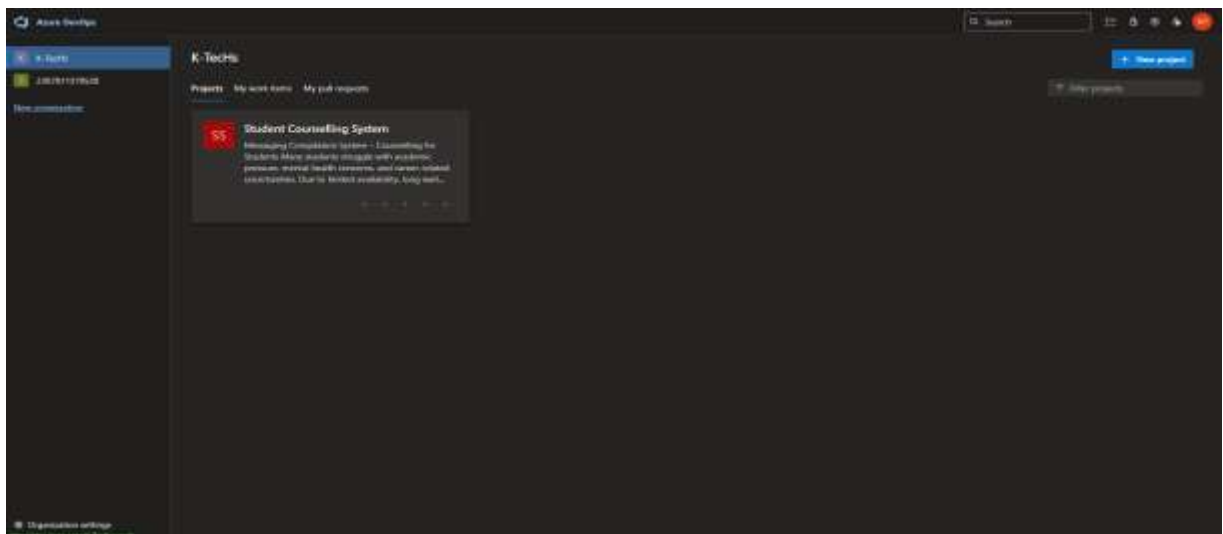
9. To manage user stories

    a.   From the left-hand navigation menu, click on Boards. This will take you to the main Boards page, where you can manage work items, backlogs, and sprints.

    b.   On the work items page, you'll see the option to Add a work item at the top. Alternatively, you can find a + button or Add New Work Item depending on the view you're in. From the Add a work item dropdown, select User Story. This will open a form to enter details for the new User Story.
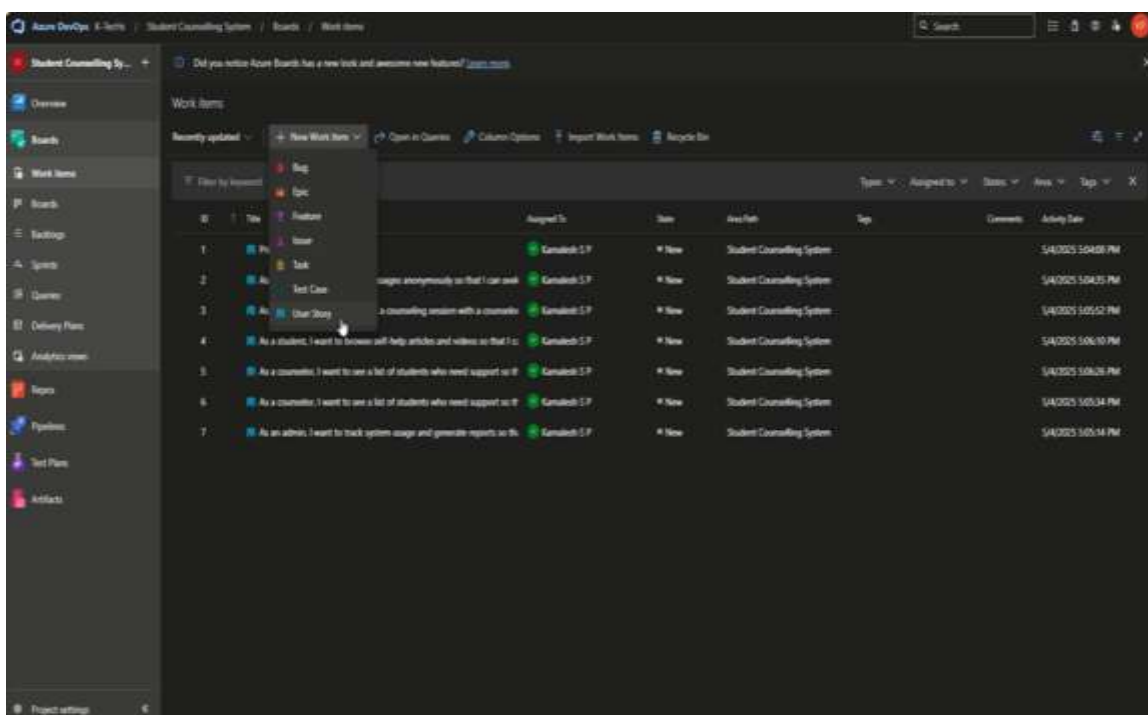
10. Fill in the User Story details
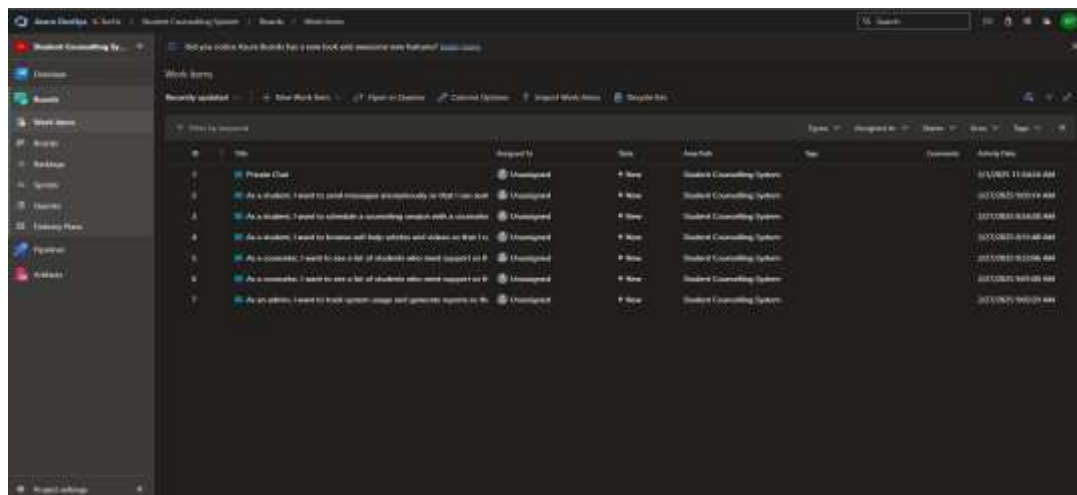
Result:

The user story was written successfully.

Ex. No. 5	DESIGNING SEQUENCE DIAGRAM

Aim:

To design a Sequence Diagram using Mermaid.js

THEORY:

A Sequence Diagram is a key component of Unified Modeling Language (UML) used to visualize the interaction between objects in a sequential order. It focuses on how objects communicate with each other over time, making it an essential tool for modelling dynamic behaviour in a system.

Procedure:

1. Open a project in Azure DevOps Organisations.

2. To design select wiki from menu
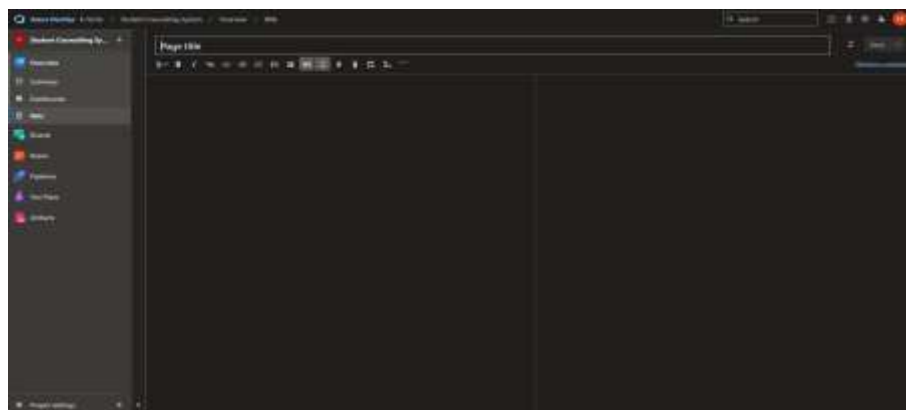
3. Write code for drawing sequence diagram and save the code.

```mermaid
::: mermaid
sequenceDiagram
participant Student
participant System
participant Counselor
participant Appointment
Student ->> System: View available slots()
System ->> Counselor: Fetch availability()
Counselor -->> System: Return available slots
System -->> Student: Display available slots
Student ->> System: Select slot & request booking()
System ->> Appointment: Create new appointment()
Appointment -->> System: Confirm appointment creation
System ->> Counselor: Notify about new appointment
Counselor ->> System: Accept or Reschedule
System ->> Student: Send confirmation notification
Student -->> System: Receive confirmation()
 :::
```

Explanation:

- Participant defines the entities involved in the sequence.
- ->> indicates a synchronous message sent from one participant to another.
- -->> indicates a return/response message.
- The sequence flows from left (Student) to right (System, Counselor, Appointment) as per interactions.
- This diagram captures the full flow of booking a counseling session:
  - Student views available slots.
  - System fetches data from Counselor, then returns options to the Student.
  - Upon selection, the System creates an Appointment, confirms it, and notifies the Counselor.
  - The Counselor either accepts or reschedules.
  - Finally, the System sends confirmation back to the Student.
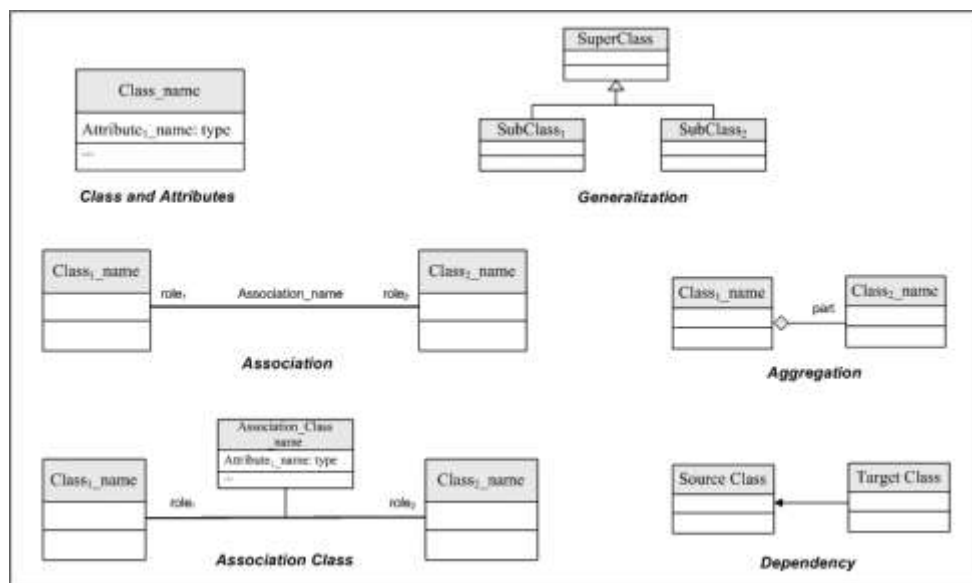
4. Click wiki menu and select the page.
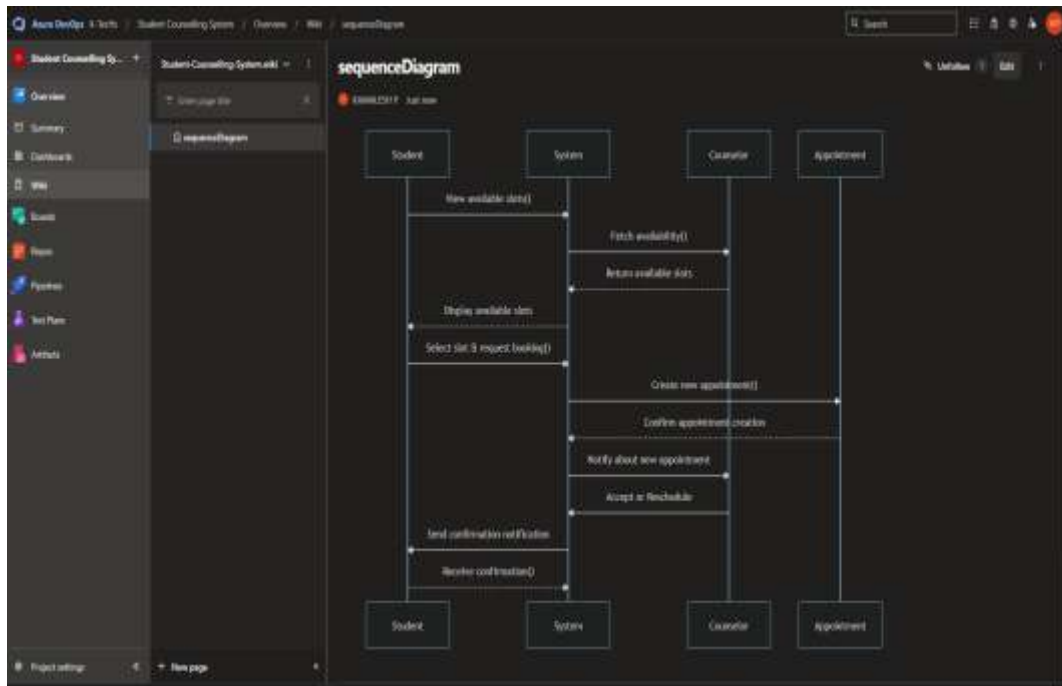
Result:

The sequence diagram was drawn successfully.

Ex. No. 6	**DESIGNING CLASS DIAGRAM**

AIM:-

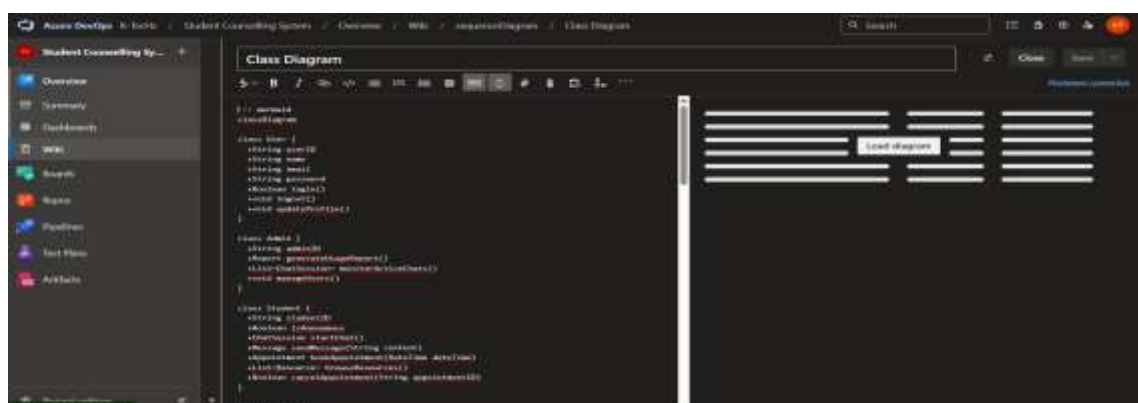To draw a sample class diagram for your project or system.

THEORY

A UML class diagram is a visual tool that represents the structure of a system by showing its classes, attributes, methods, and the relationships between them.

Notations in class diagram

Procedure:

      1. Open a project in Azure DevOps Organisations.

      2. To design select wiki from menu

3. Write code for drawing class diagram and save the code

Code:

```mermaid
::: mermaid
classDiagram

class User {
  +String userID
  +String name
  +String email
  +String password
  +Boolean login()
  +void logout()
  +void updateProfile()
}

class Admin {
  +String adminID
  +Report generateUsageReport()
  +List~ChatSession~ monitorActiveChats()
  +void manageUsers()
}

class Student {
  +String studentID
  +Boolean isAnonymous
  +ChatSession startChat()
  +Message sendMessage(String content)
  +Appointment bookAppointment(DateTime dateTime)
```

15

```
    +List~Resource~ browseResources()

    +Boolean cancelAppointment(String appointmentID)

}


class System {

    +void sendNotification(String userID, String message)

    +void assignCounselor(String chatID, Counselor)

    +String estimateWaitTime()

}


class Report {

    +String reportID

    +String generatedBy

    +DateTime timestamp

    +String reportType

    +Boolean exportToPDF()

    +Boolean exportToCSV()

}


class ChatSession {

    +String chatID

    +Boolean isAnonymous

    +List~Message~ messages

    +DateTime startTime

    +DateTime endTime

    +void addMessage(User sender, String content)

    +void endChat()

}


class Message {
```

16

```
    +String messageID

    +String senderID

    +DateTime timestamp

    +String content

    +File file

    +void editMessage(String content)

    +Boolean deleteMessage()

}


class Resource {

    +String resourceID

    +String title

    +String category

    +String contentURL

    +String getResourceDetails()

    +List~Resource~ filterResources(String category)

}


class Counselor {

    +String counselorID

    +String specialization

    +List~ChatSession~ chatRequests

    +List~ChatSession~ viewChatRequests()

    +Boolean acceptChatRequest(String chatID)

    +Boolean declineChatRequest(String chatID)

    +Message sendMessage(String chatID, String content)

    +List~ChatSession~ viewStudentHistory(String studentID)

}


class Appointment {
```

```
    +String appointmentID

    +String studentID

    +String counselorID

    +DateTime appointmentTime

    +String status

    +Boolean confirmAppointment()

    +Boolean cancelAppointment()

    +Boolean rescheduleAppointment(DateTime newDateTime)

}


%% Relationships

User <|-- Admin

User <|-- Student

Student --> ChatSession : initiates

Student --> Message : sends

Student --> Appointment : books

Student --> Resource : accesses

Admin --> Report : generates

Admin --> ChatSession : monitors

Report --> ChatSession : belongsTo

ChatSession --> Message : contains

Message --> ChatSession : linkedTo

ChatSession --> Counselor : assignedTo

ChatSession --> Appointment : mayLeadTo

Counselor --> ChatSession : manages

Counselor --> Message : sends

Appointment --> Counselor : scheduledWith

Appointment --> Student : bookedBy

System --> Counselor : assigns

System --> Student : notifies
```

Resource --> Student : usedBy

:::

Relationship Types

Type	Description
<|	Inheritance
*	Composition
o	Aggregation
>	Association
<	Association
|>	Realization
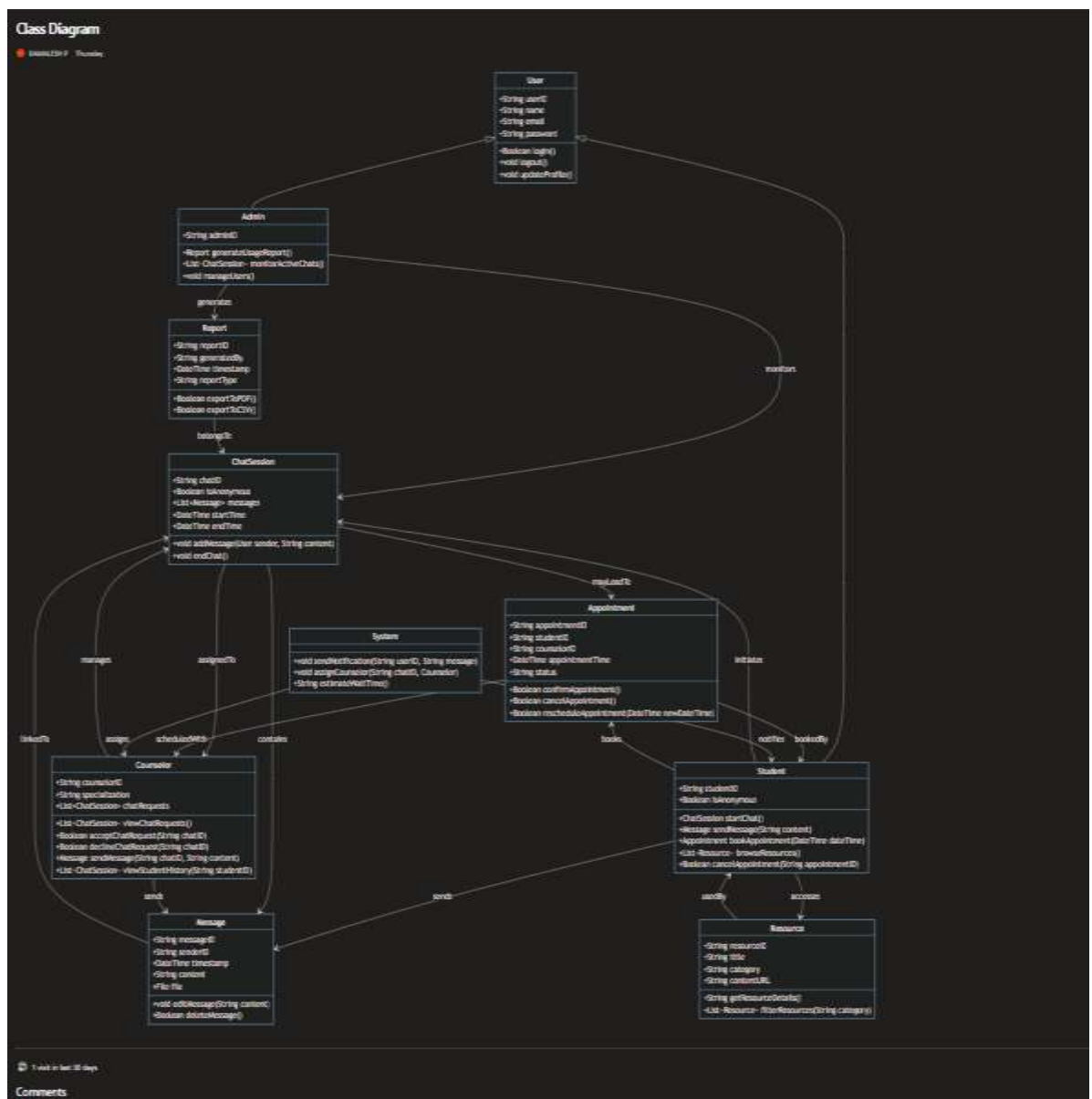
**Result:**

The use case diagram was designed successfully.

19

Ex. No. 7	USECASE DIAGRAM

Aim:

Steps to draw the Use Case Diagram using PlantUML Web Editor

Theory:

♦ UCD shows the relationships among actors and use cases within a system which Provide an overview of all or part of the usage requirements for a system or organization in the form of an essential model or a business model and communicate the scope of a development project

- **Use Cases**

- **Actors**

- **Relationships**

- **System Boundary Boxes**
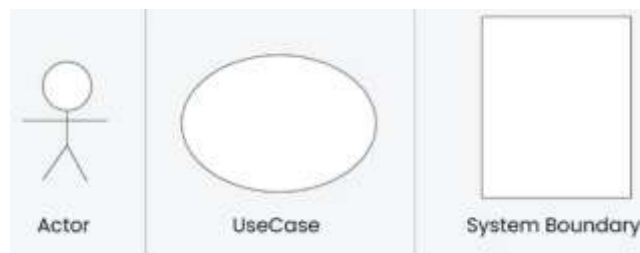
Procedure

Stage 1:
Step 1: Create the Use Case Diagram in PlantUML Web Editor
- Open the PlantUML Web Editor (https://editor.plantuml.com/).
- Paste your PlantUML code into the editor.
- Use @startuml and actor, usecase keywords to define actors and use cases.
- Assign aliases to each use case (e.g., "Send Automated Notifications" as UC1).
- Connect actors to use cases using arrows (actor --> usecase).
- Verify the visual layout on the right-hand side of the screen.
- Click **Save**, **Copy**, or use **SyncURL** to get a sharable link for integration (e.g., Azure).

Step 2: Export or Screenshot the Diagram
- Use browser tools to take a screenshot or right-click on the diagram to save it as an image.

- Ensure the image is clear and labeled properly for use in documentation or reports.

Step 3: Add Diagram to Report or Azure Platform
- Paste the image or URL into your report section.
- Clearly title the diagram as **"Use Case Diagram - Counseling Support System"** or as per your project name.
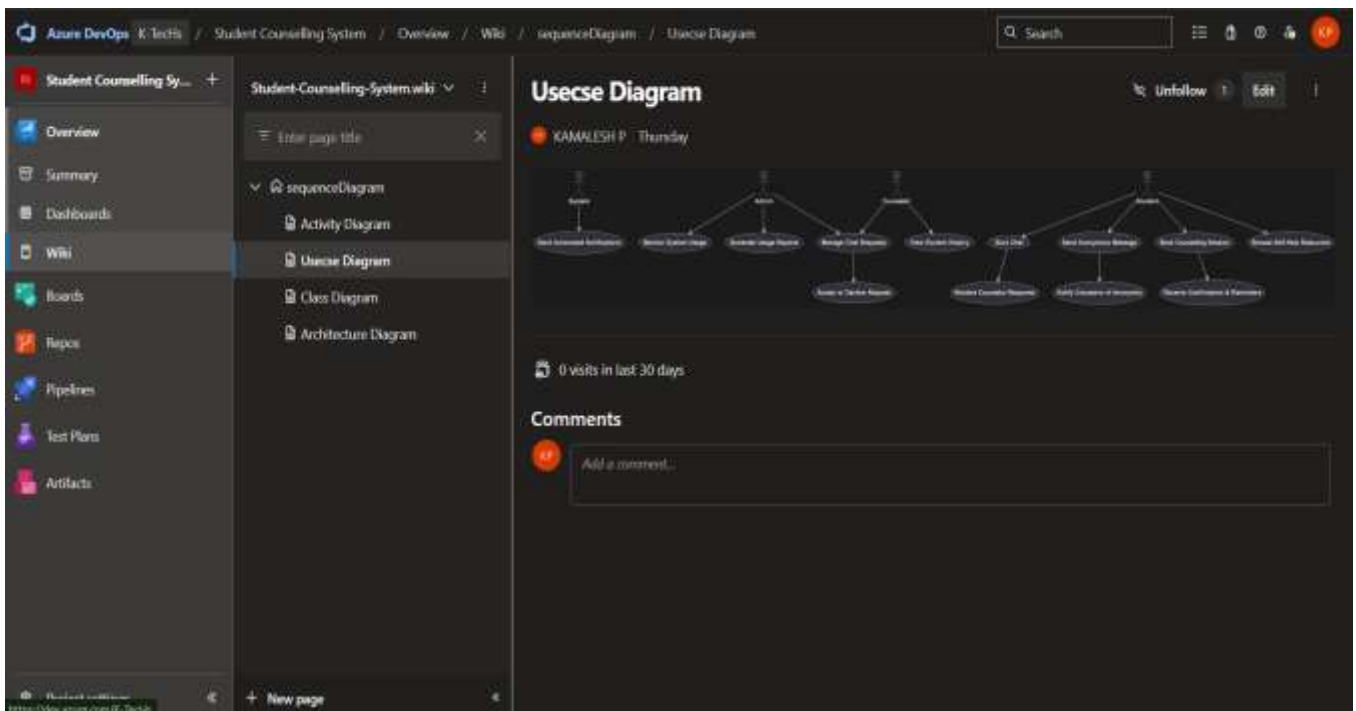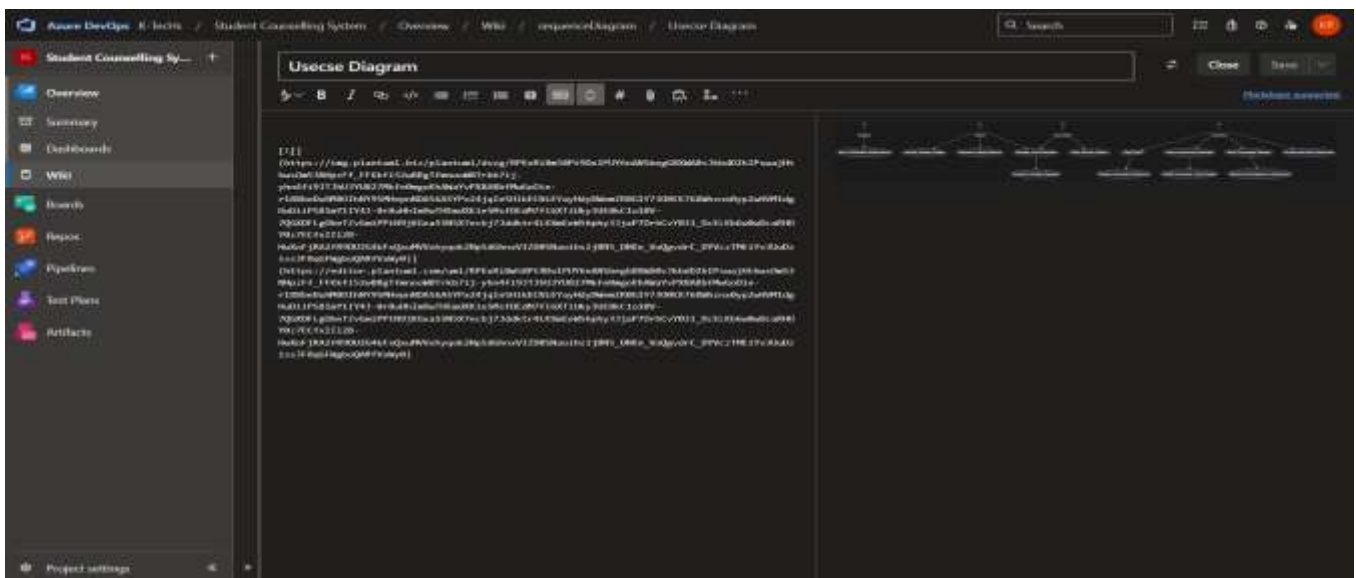- Briefly explain actor interactions if necessary.
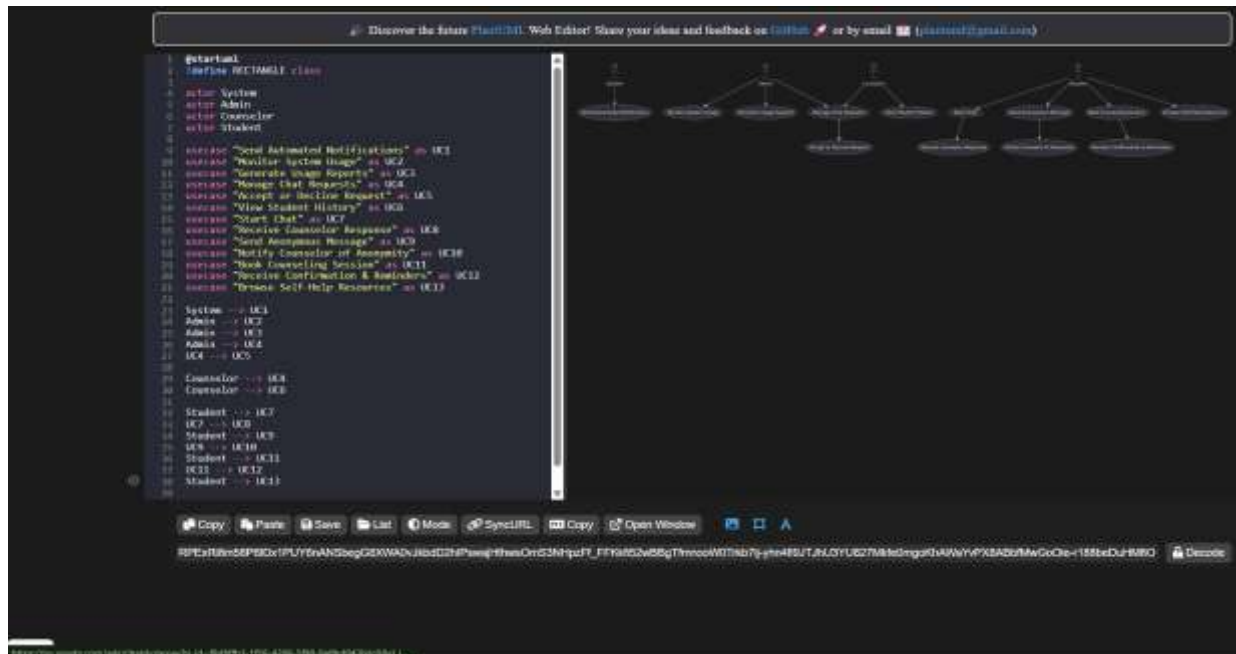
Stage 2:
Step 1: Upload the Diagram to Azure DevOps

Option 1: Add to Azure DevOps Wiki using PlantUML Link
- Open Azure DevOps and go to your project.
- Navigate to **Wiki** (Project > Wiki).
- Click **"Edit Page"** or create a new page.
- Paste the **PlantUML SyncURL link** directly into the content area.
- Make sure link recognition is enabled so the link becomes clickable automatically.
- Optionally, add a brief description below the link to explain the use case diagram.

Option 2: Attach to Work Items in Azure Boards

- Open Azure DevOps → Go to **Boards** (Project > Boards).
- Select the relevant **User Story**, **Task**, or **Feature**.
- Click on the **"Discussion"** or **"Attachments"** tab.
- Paste the **PlantUML SyncURL link** directly.
- Add a comment or description to explain the diagram's context or functionality.

Usecse Diagram

[![]
(https://img.plantuml.biz/plantuml/dsvg/RPVxRi8wS8Pv9Dx1PUYVxwAXSmg1B0q6Rv3bnXQ2k2Puuujin
bacOwSJ0QxcF_FFEkf1CUuBFgTfmxsnWBTr6nb7rj.
ybn1fz4l5TJHdrJYUB27MbFmMmgoXx9kzYvFXBX8bFMaSnZkx-
r1XBbsDw9RRtDt8Y95PXhqpzXDE5iK5YPxz24JqZxFUt1kE3kLfYayHkpUSuwoDXHC2Y7X0HKXTkRwhxcosNyyzSu9HM1uJg
RsXLiPSR5wT1IY43-9r0uAkImRwSHSxw0X1rSRm40Sp0FwYY9X9iXWYT1k4k42T80k0vXYIY3_DkXkXh6kwBwoSLw0XRI.
7Q5O0RLgObsrTZv0aS3PPH0RtkXisa3URUX7kxkj73hkr4L4XbmxXsphkaqhtyR3juFYDrwCxYRI1kUSfXkUbmjhwSkw0RmB
YRiYYXfxzk22LQD-
HwXoFsRAZVR9XX3Zk6kuFslQksxMV5kUyxpXS2RpCxDihnnxVX20WXUkoibm3JNYL_SM4k-2VADcowoF3Pm9DvY8nXNX3]]
(https://editor-plantuml.com/uml/RPPxaRLBwSXOPLXRhxlPUYYksXXSknqLmTXXaBnXN1WgRk9Y9X9k92kXXPnnjXXXHwySAwXXi3
RWxIRPT-FTKX4lC3nXiBgTfmxsnWXBTrXXk7kj.ybn4Fl5YL5lTXY5SIXZSXMnXmbkgoUXDkpHX9YkXDXkXXXXaXiXbXz20ki2e-
r1XBbsDw9RRtDt8Y95PXhqpzXDE5iK5YPxz24JqZxFUt1kE3kLfYYayHkpUSuwoDXXC2Y7X0X0kXYtXkwwhkXcosNyyzSu9HM1uJg
RsXLiPSR5wT1IY43-9r0uAkImRwSHSxw4X1kSkX9kkXHU4XTkXXT1kXkpSY5XXX1kXX0V-
7Q5O0RLgObsrTXv0aS3PPH0RtkXisa3URUX7kxkj73hkhr1kXLKXBmkwXsphXaqhkXj3juFYDrkXYX1_XkXY0XXUkwSwXikwSZw0RmX
YRiYYXfkxz22LQX-
HwXoFsRAZVX9XX3Zk6kZsuXfVXkxypak2RpSdXkvoYX20WXSksxTn1JjMYL_SM4kx_WkQpsXvC_DYYcxTWkzXwXXdXz
ixx3fXkqUsHkqkuQWYYXsWyXj]

Ex. No. 8	ACTIVITY DIAGRAM

AIM:

To draw a sample activity diagram for your project or system.

THEORY

Activity diagrams are an essential part of the Unified Modelling Language (UML) that help visualize workflows, processes, or activities within a system. They depict how different actions are connected and how a system moves from one state to another.

Notations	Symbol	Meaning
Start	●	Shows the beginning of a process
Connector	→	Shows the directional flow, or control flow, of the activity
Joint symbol	▬	Combines two concurrent activities and re-introduces them to a flow where one activity occurs at a time
Decision	◇	Represents a decision
Note	⬜	Allows the diagram creators o communicate additional messages
Send signal	⬠	Show that a signal is being sent to a receiving activity
Receive signal	⬡	Demonstrates the acceptance of an event
Flow final symbol	⊗	Represents the end of a specific process flow
Option loop	⬓	Allows the creator to model a repetitive sequence within the option loop symbol
Shallow history pseudostate	Ⓗ	Represents a transition that invokes the last active state.
End	◉	Marks the end state of an activity and represents the completion of all flows of a process

Procedure:

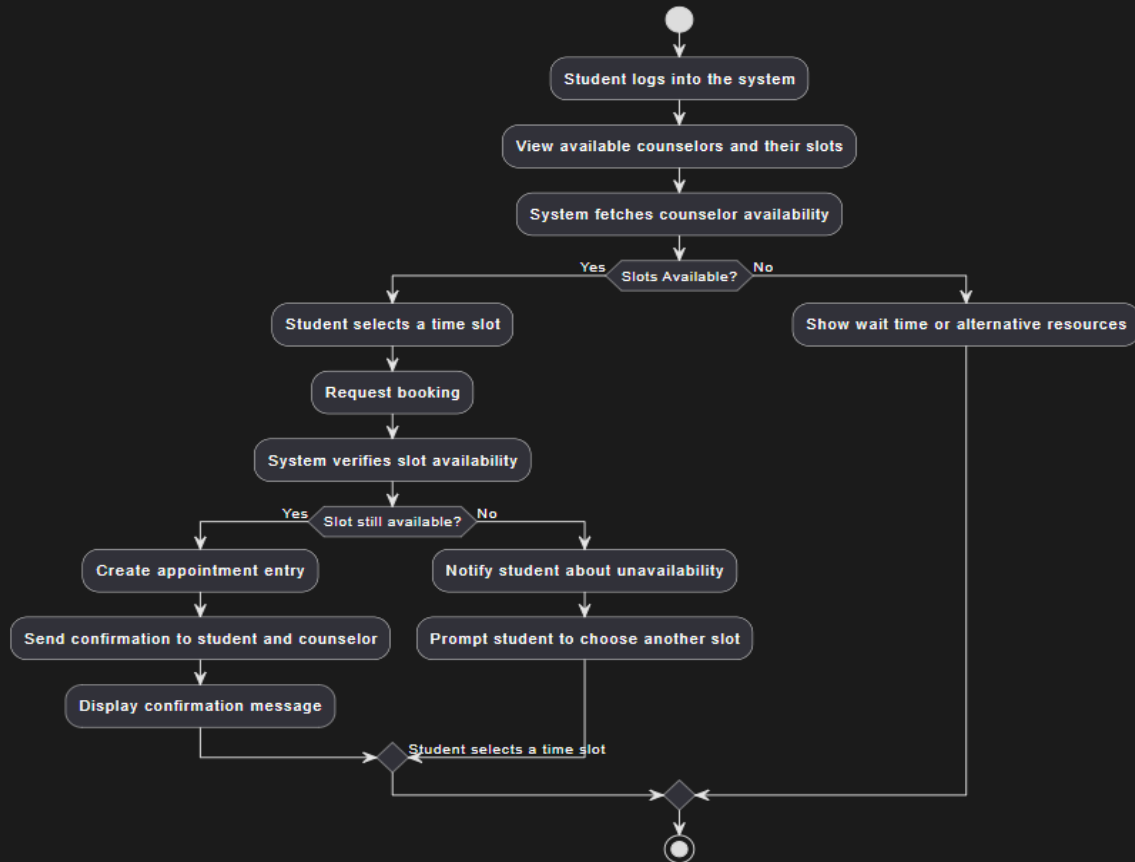1. Create the Use Case Diagram in PlantUML Web Editor.
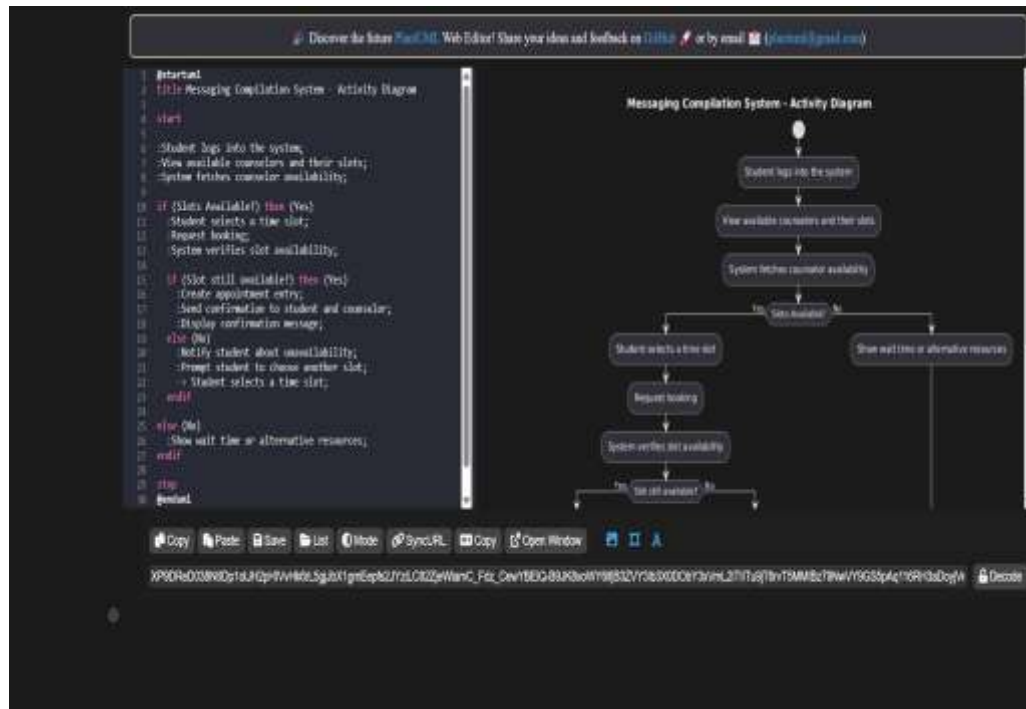
2. Upload the diagram in Azure DevOps wiki.
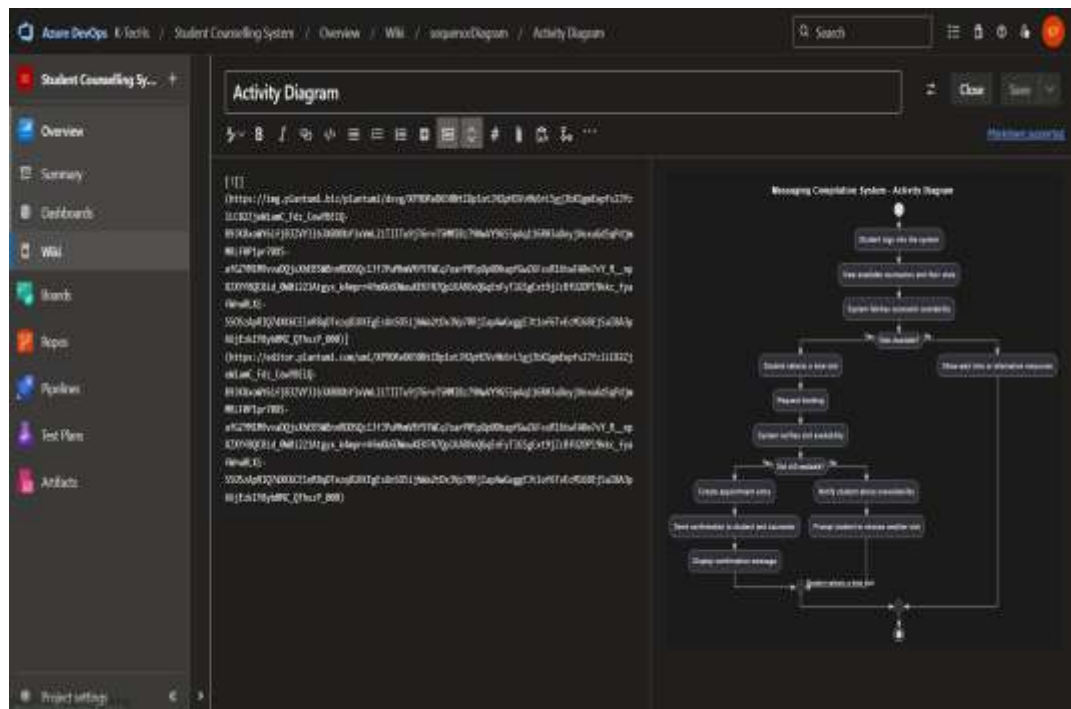
Activity Diagram

Messaging Compilation System - Activity Diagram

Result:

The activity diagram was designed successfully

25

Ex. No. 9	**ARCHITECTURE DIAGRAM**

Aim:

Steps to draw the Architecture Diagram using PlantUML Web Editor.

Theory:

An architectural diagram is a visual representation that maps out the physical implementation for components of a software system. It shows the general structure of the software system and the associations, limitations, and boundaries between each element.
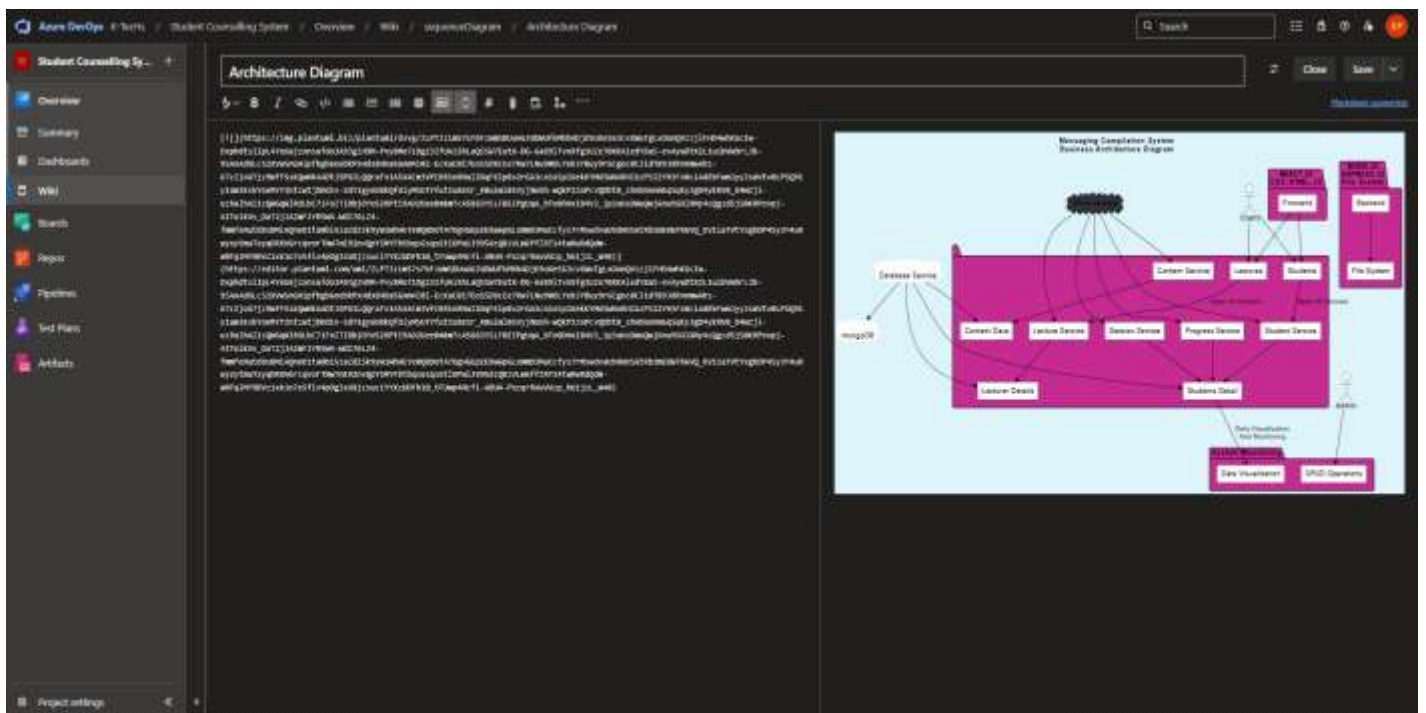
Procedure:

1. Create the Use Case Diagram in PlantUML Web Editor.

2. Upload the diagram in Azure DevOps wiki
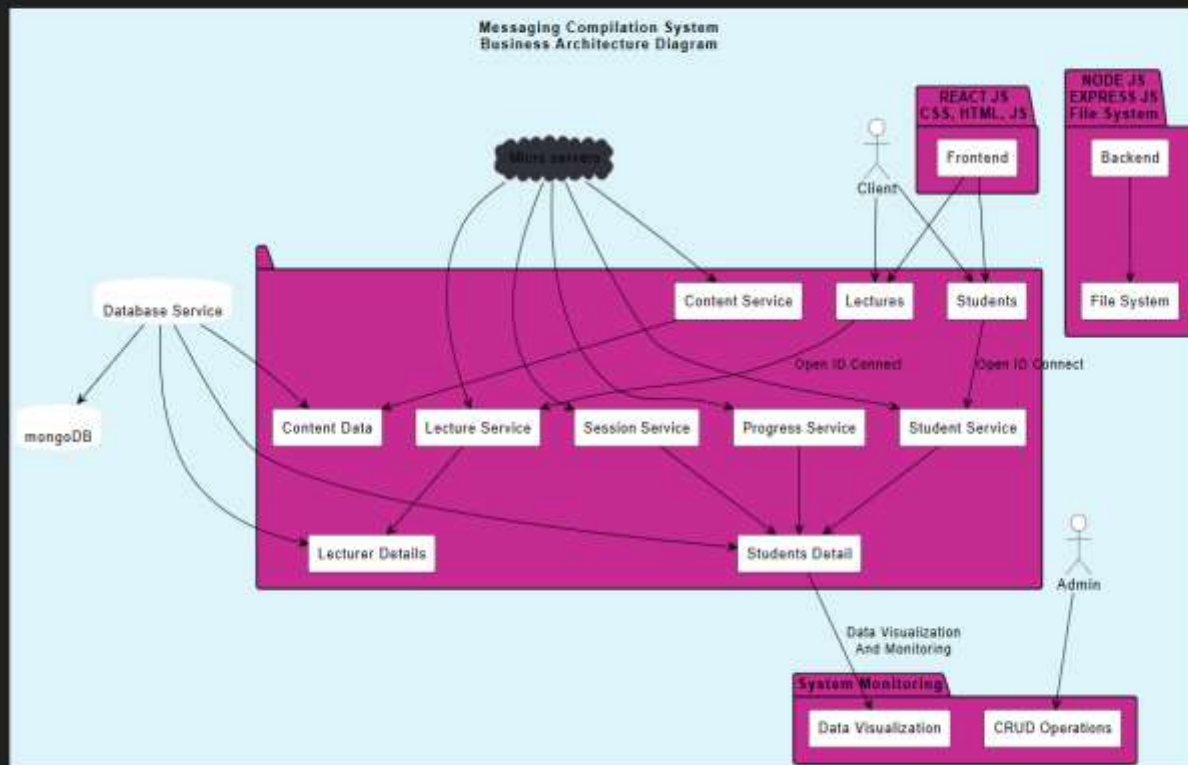
KAMALESH P Just now

Messaging Compilation System
Business Architecture Diagram
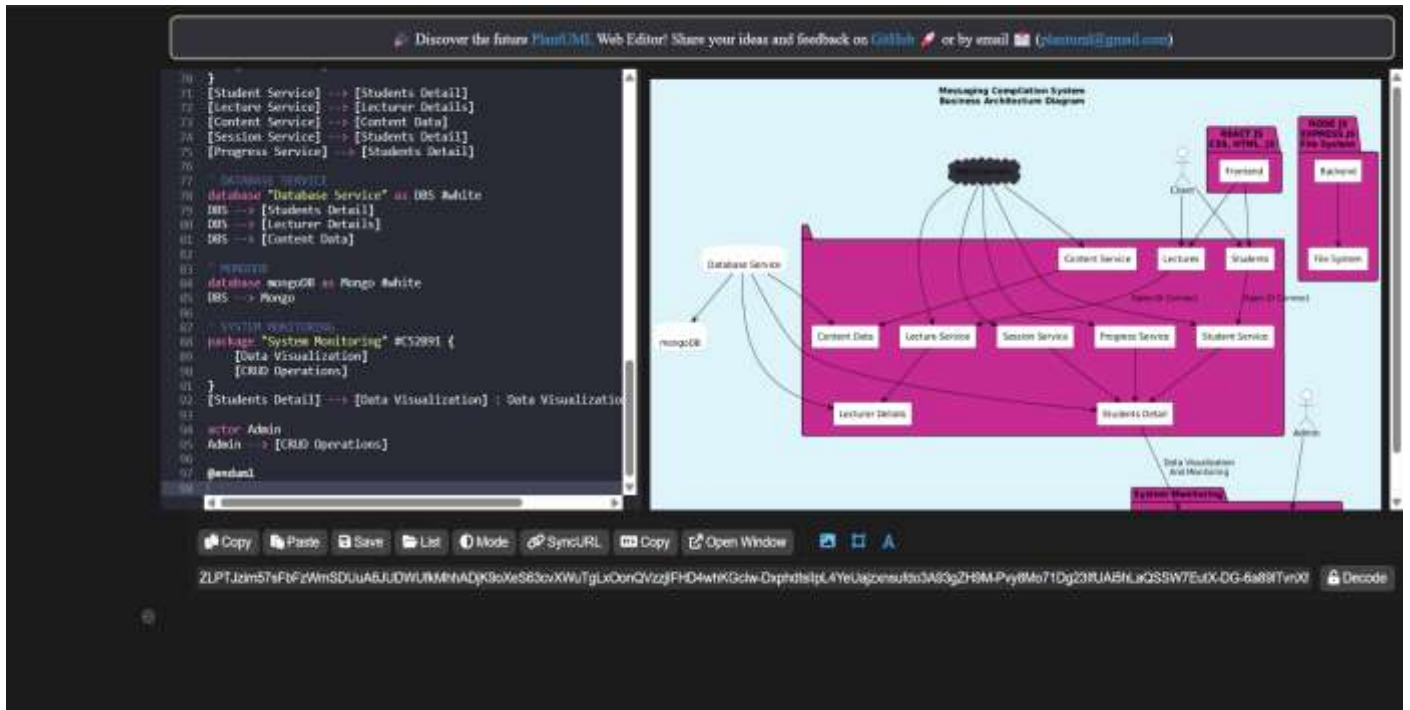
Result:

The architecture diagram was designed successfully

28

Ex. No. 10	**USER INTERFACE**

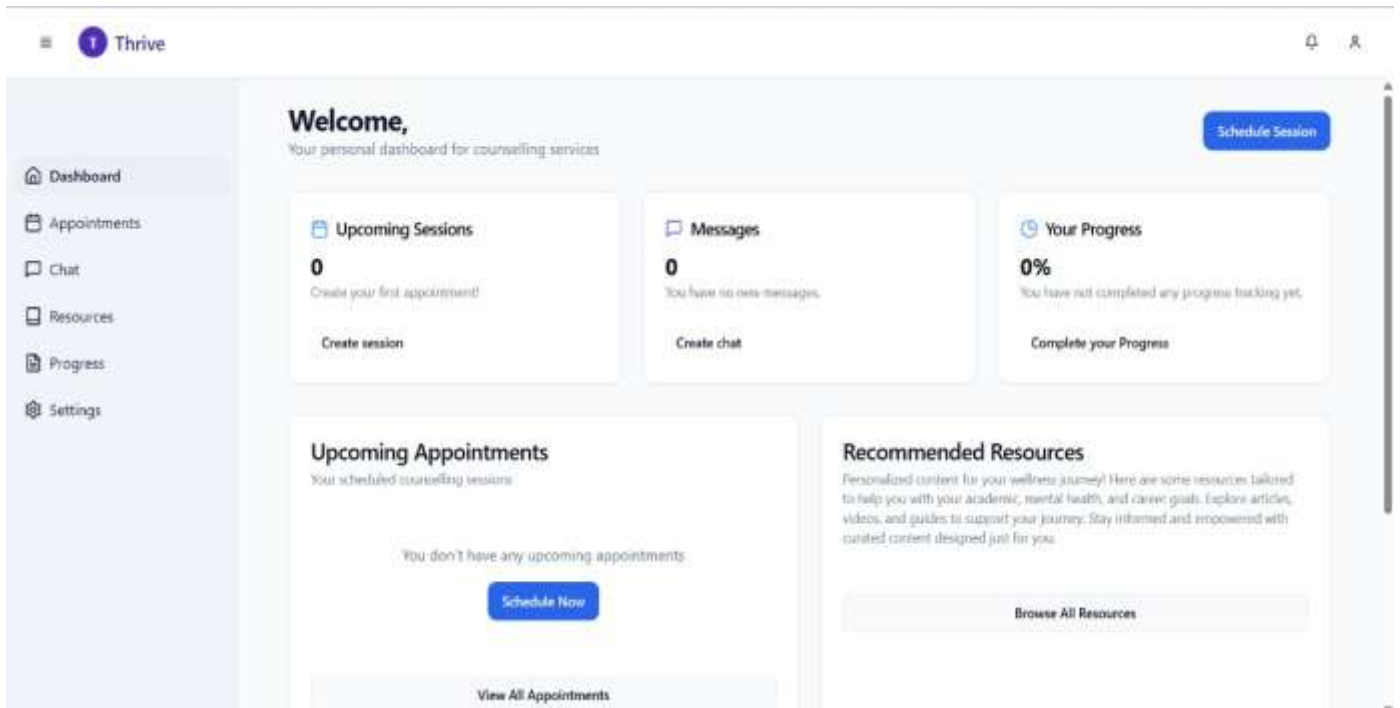**Aim:**

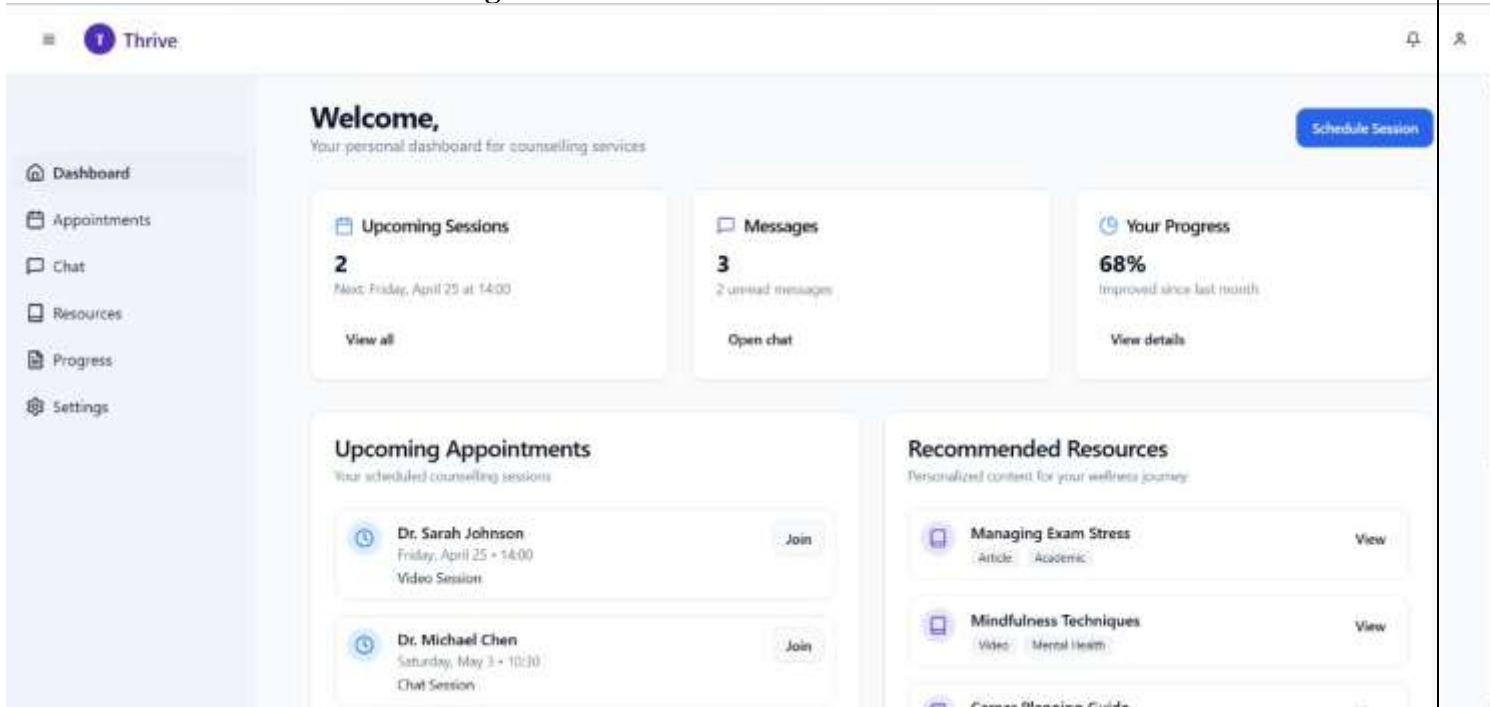To Design User Interface for the given project.

UI:

1. VS Code Screen Shoot:
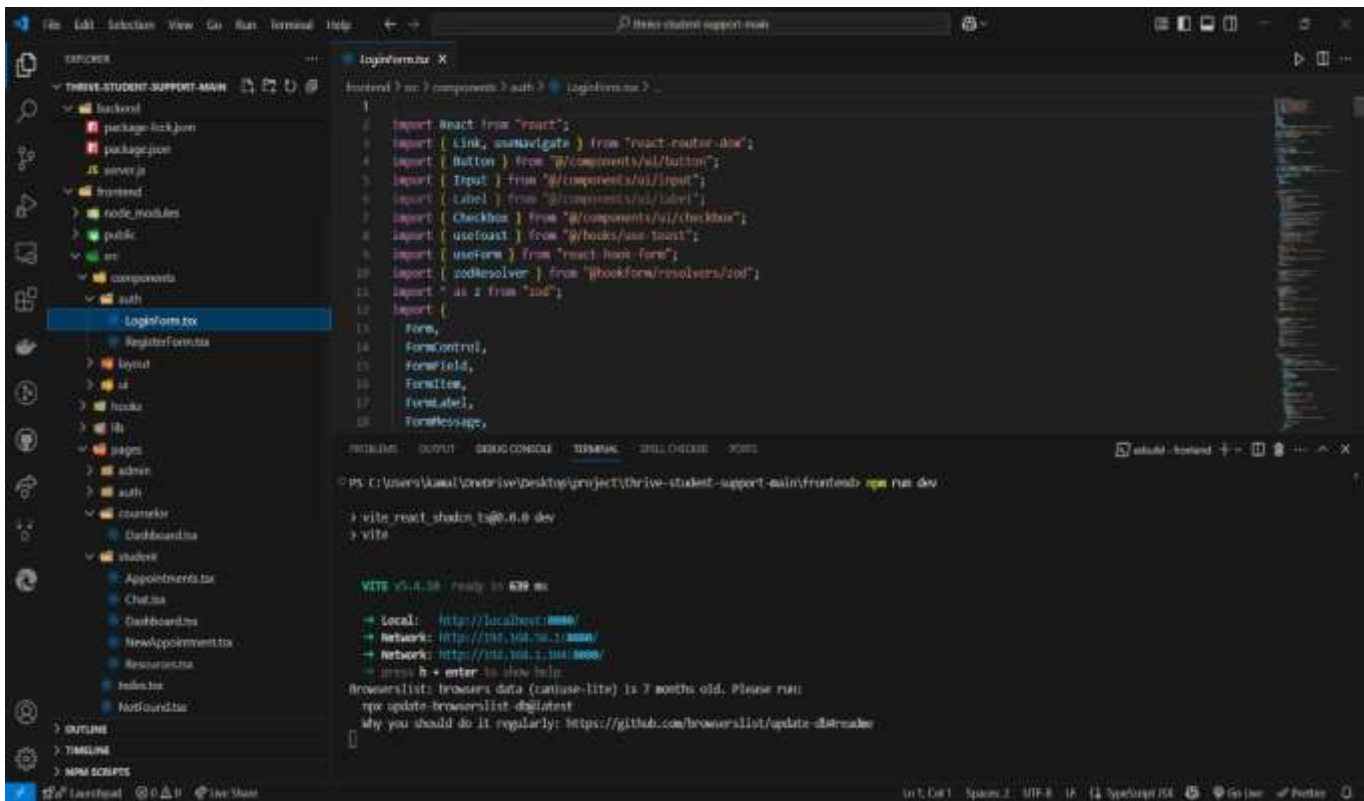
2. UI for New User:

3. UI for Existing Users:

Result:

The UI was designed successfully.

<table>
<tr><td>

</td><td>

IMPLEMENTATION

</td></tr>
</table>

Aim:

To implement the given project based on Agile Methodology.

Procedure:

Step 1: Set Up an Azure DevOps Project
- Log in to Azure DevOps.
- Click "New Project" → Enter project name → Click "Create".
- Inside the project, navigate to "Repos" to store the code.

Step 2: Add Your Web Application Code
- Navigate to Repos → Click "Clone" to get the Git URL.
- Open Visual Studio Code / Terminal and run:

        git clone <repo_url>
        cd <repo_folder>

- Add web application code (HTML, CSS, JavaScript, React, Angular, or backend like Node.js, .NET, Python, etc.).
- Commit & push:

        git add .
        git commit -m "Initial commit"
        git push origin main

Step 3: Set Up Build Pipeline (CI/CD - Continuous Integration)
- Navigate to Pipelines → Click "New Pipeline".
- Select Git Repository (Azure Repos, GitHub, or Bitbucket).
- Choose Starter Pipeline or a pre-configured template for your framework.
- Modify the azure-pipelines.yml file (Example for a Node.js app):

```
trigger:
 - main

pool:
 vmImage: 'ubuntu-latest'

steps:
 - task: UseNode@1
  inputs:
    version: '16.x'

 - script: npm install
   displayName: 'Install dependencies'

 - script: npm run build
```

```
        displayName: 'Build application'

     - task: PublishBuildArtifacts@1
       inputs:
         pathToPublish: 'dist'
         artifactName: 'drop'
```

Click "Save and Run" → The pipeline will start building app.

## Step 4: Set Up Release Pipeline (CD - Continuous Deployment

- Go to Releases → Click "New Release Pipeline".
- Select Azure App Service or Virtual Machines (VMs) for deployment.
- Add an artifact (from the build pipeline).
- Configure deployment stages (Dev, QA, Production).
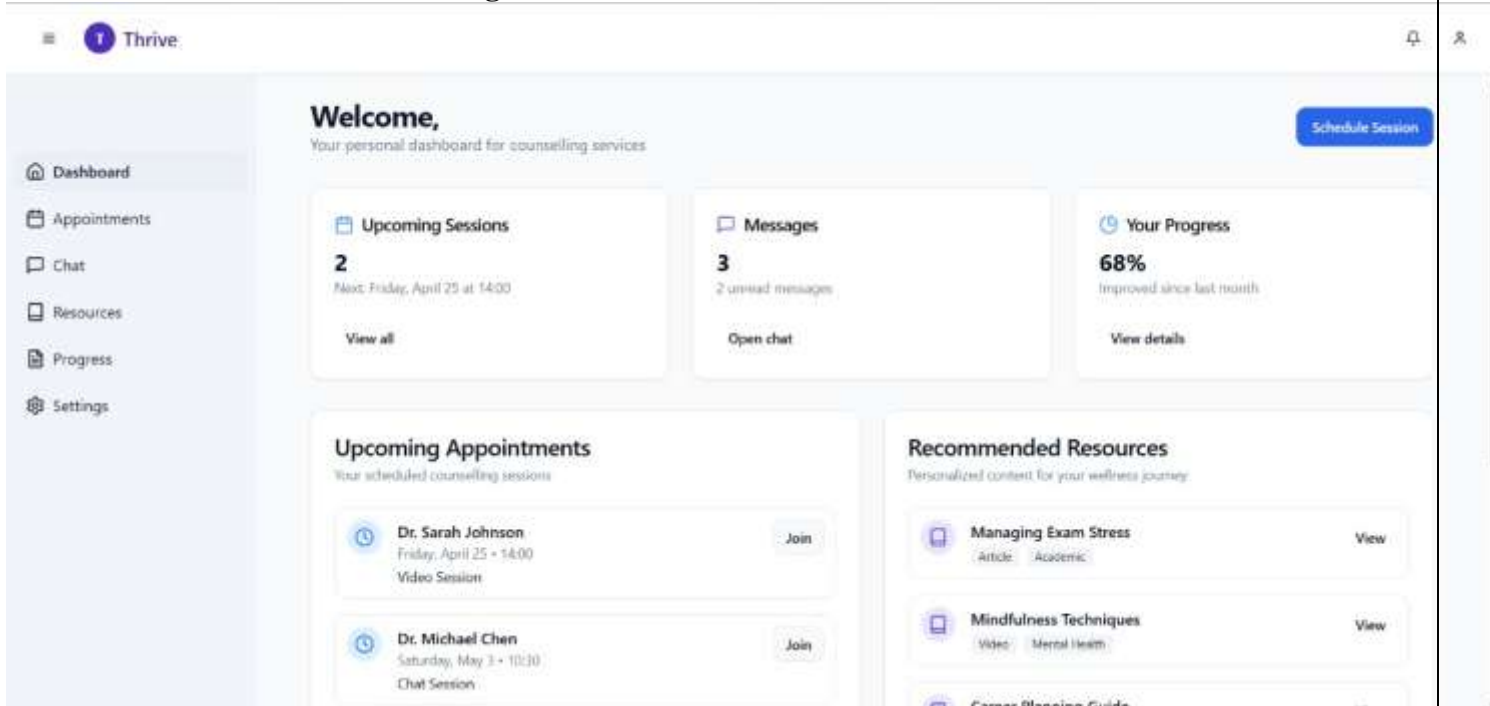- Click "Deploy" to push your web app to Azure.

Implementation Screen Shots:

1. CODE IMPLEMENTTION IN AZURE DevOps:
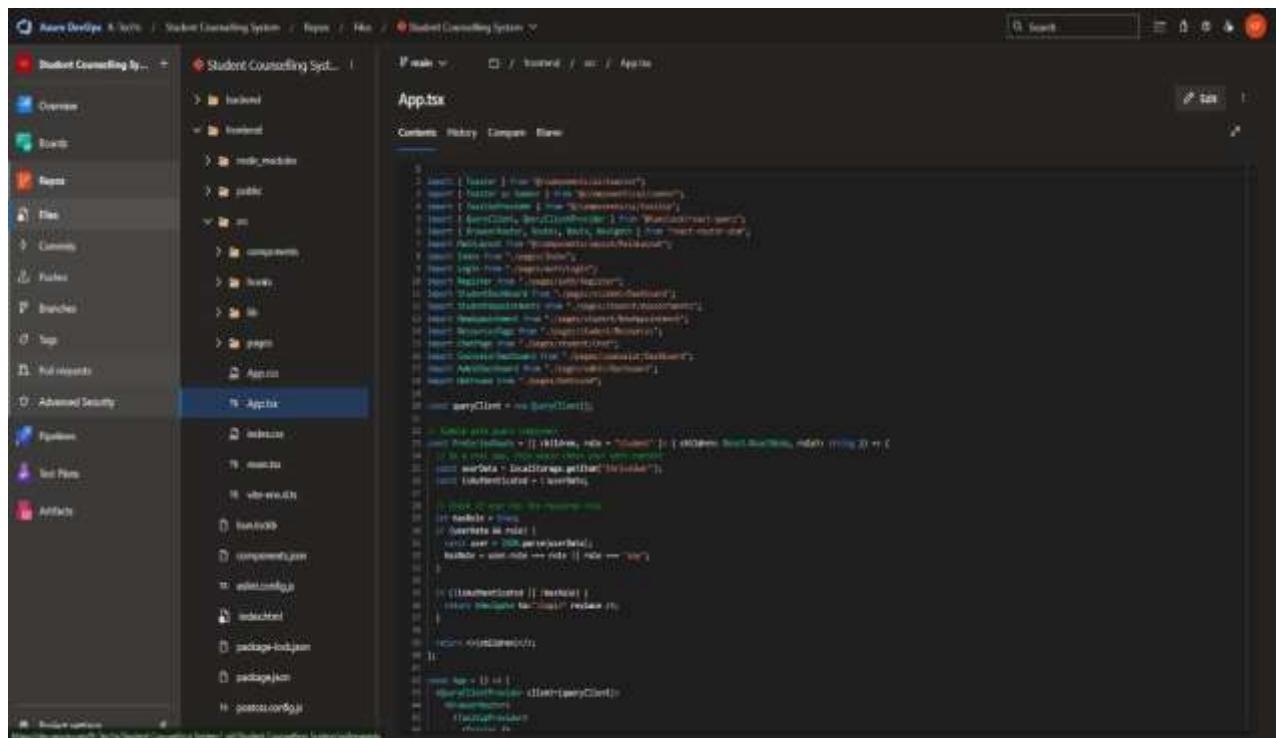
2. I for New User:
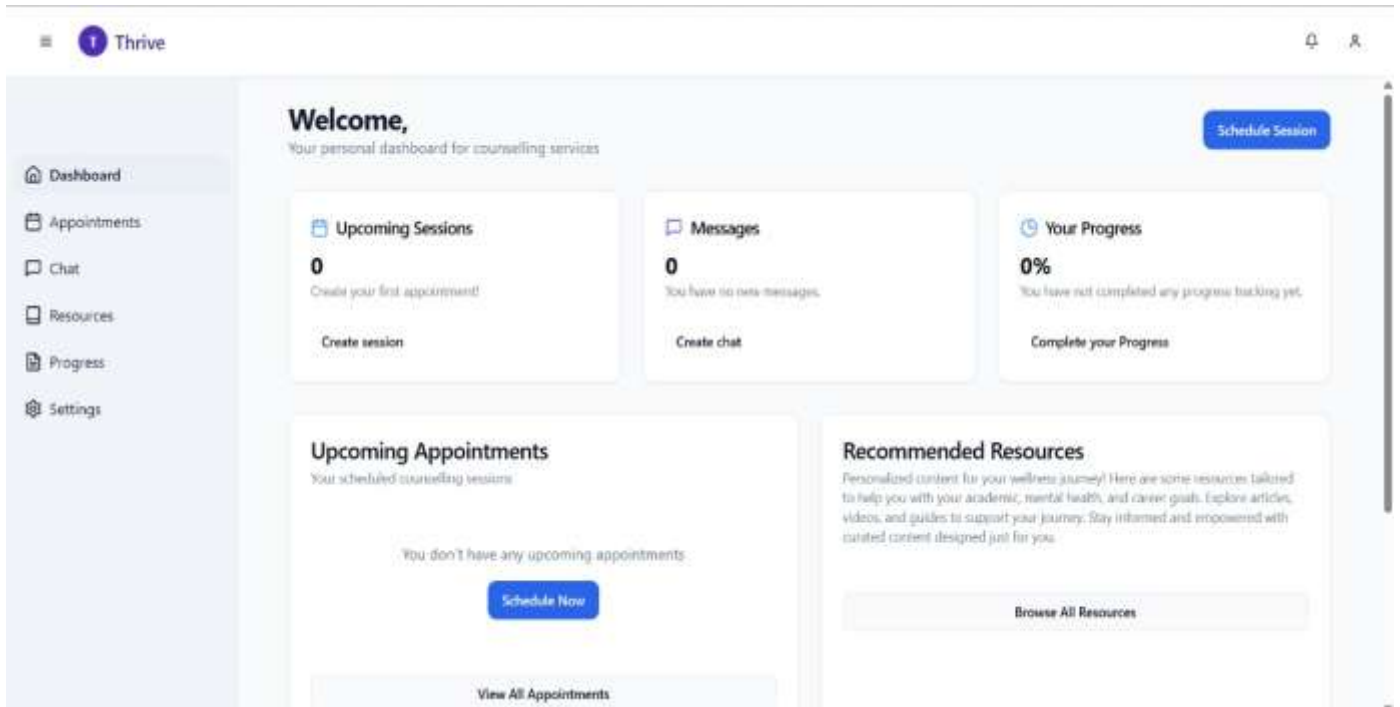
3. UI for Existing Users:

Result

        Thus the application was successfully implemented.