RAJALAKSHMI ENGINEERING COLLEGE

An AUTONOMOUS Institution

Affiliated to ANNA UNIVERSITY, Chennai

DEPARTMENT OF COMPUTER SCIENCE &

ENGINEERING ACADEMIC YEAR 2024-2025

EVEN SEMESTER

CS23432 SOFTWARE ENGINEERING LAB

LAB MANUAL

SECOND YEAR

FOURTH SEMESTER

2024- 2025

EVEN SEMESTER

Ex No	List of Experiments
1	Study of Azure DevOps
2	Designing Project using AGILE - SCRUM Methodology.
3	Agile Planning
4	User stories – Creation
5	Architecture Diagram Using AZURE
6	Designing Use case and Class Diagram
7	Designing Interaction Diagrams
8	Design Interface
9	Implementation – Design a Web Page based on Scrum Methodology
10	Testing using Azure.
11	Deployment

Requirements	
Hardware	Intel i3, CPU @ 1.20GHz 1.19 GHz, 4 GB RAM, 32 Bit Operating System
Software	StarUML , Azure

LAB PLAN

CS19442-SOFTWAREENGINEERINGLAB

Ex No	Date	Topic	Page No	Sign
1		Studyof AzureDevOps		
2		WritingProblemStatement		
3		Designing Project using AGILE/SCRUM Methodology by using Azure.		
4		Agile Planning		
5		User stories – Creation		
6		ArchitectureDiagramUsing AZURE		
7		DesigningUsecase Diagramusing StarUML		
8		DesigningActivityDiagramsusing StarUML		
9		DesigningSequenceDiagramsusing StarUML		
10		DesignClass Diagram		
11		DesignUser Interface		
12		Implementation. Design a Web Page based on Scrum Methodology		
13		Testing		
14		Deployment		

Course Outcomes (COs)

CourseName: SoftwareEngineering

CourseCode: CS23432

CO 1	Understand the software development process models.
CO 2	Determine the requirements to develop software
CO 3	Apply modeling and modeling languages to design software products Apply
CO 4	various testing techniques and to build a robust software products
CO 5	Manage Software Projects and to understand advanced engineering concepts

CO - PO – PSO matrices of course

PO/PSO / CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CS23432.1	2	2	3	2	2	2	2	2	2	2	3	2	1	3	-
CS23432.2	2	3	1	2	2	1	-	1	1	1	2	-	1	2	-
CS23432.3	2	2	1	1	1	1	1	1	1	1	1	1	2	2	1
CS23432.4	2	2	3	2	2	2	1	0	2	2	2	1	1	2	1
CS23432.5	2	2	2	1	1	1	1	0	2	1	1	1	2	1	-
Average	2.0	2.2	2.0	1.6	1.6	1.4	1.3	1.3	1.6	1.4	1.8	1.3	1.4	2.0	1.0

Correlation levels 1, 2 or 3 are as defined below:
1: Slight (Low) 2: Moderate (Medium) 3: Substantial (High) No correlation: "-"

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

PROBLEM STATEMENT

AIM:

To prepare PROBLEM STATEMENT for your given project.

Problem Statement: Music Playlist Batch Creator

With the growing demand for personalized music experiences and digital content monetization, there is a need for a system that enables users to efficiently manage, organize, and access music files in bulk. The goal is to develop a Music Playlist Batch Creator application that allows users to upload multiple music files at once, automatically categorize songs based on metadata (e.g., genre, artist, mood), and generate personalized playlists according to user-defined preferences. The system should also include a secure monetization feature, where songs can only be downloaded after successful payment, ensuring fair compensation for content creators. To enhance metadata extraction and playlist intelligence, the application should integrate with music metadata APIs (e.g., Spotify, Last.fm, MusicBrainz). Furthermore, efficient data structures such as linked lists or trees should be explored for organizing and navigating playlists, offering flexibility in sorting and retrieval operations.

Key Objectives:

1. Enable users to batch-upload multiple audio files.
2. Extract and organize metadata (artist, genre, etc.) using integrated APIs.
3. Provide automated playlist generation based on user preferences.
4. Implement payment integration before allowing downloads of songs.
5. Use appropriate data structures (linked lists, trees) for playlist management and browsing.

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

CREATE USER STORIES

Aim:

To create User Stories

THEORY

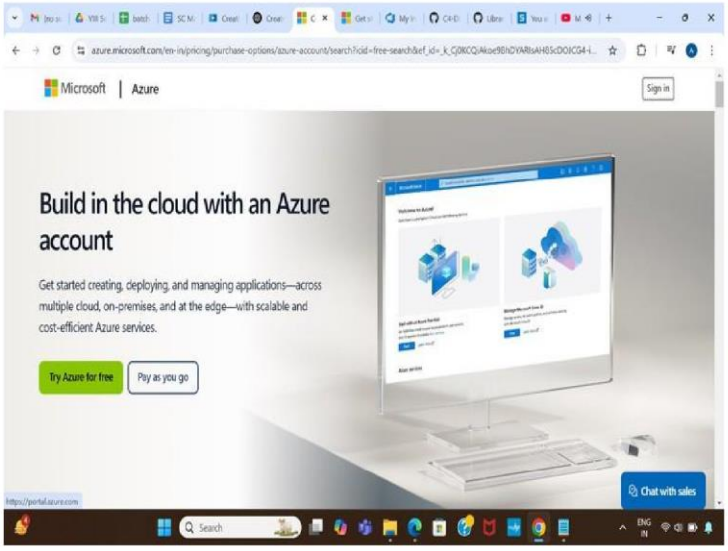
A user story is an informal, general explanation of a software feature written from the perspective of the end user. Its purpose is to articulate how a software feature will provide value to the customer.

User story template

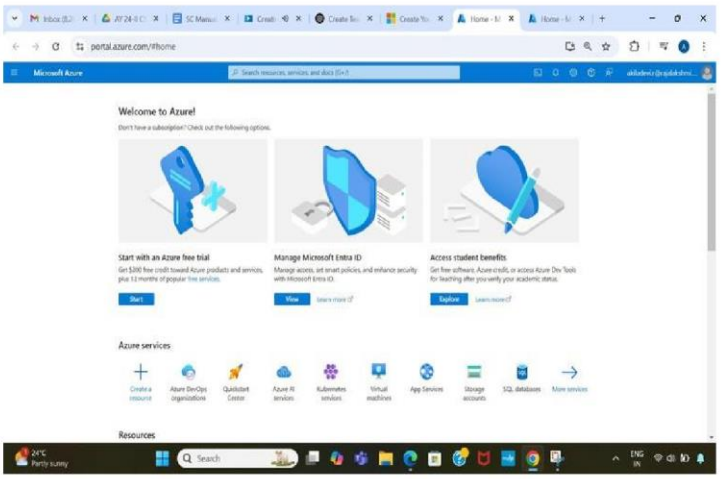
"As a [role], I [want to], [so that]."

Procedure:

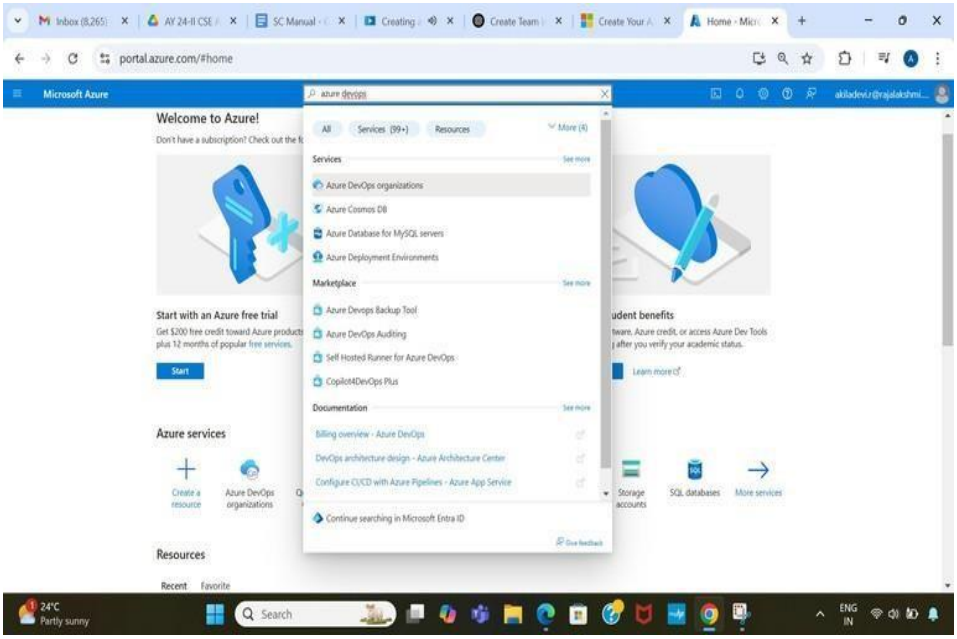
1. Open your web browser and go to the Azure website: *https://azure.microsoft.com/en-in* Sign in using your Microsoft account credentials. If you don't have an account, you'll need to create one.
2. If you don't have a Microsoft account, you can sign up for *https://signup.live.com/?lic=1*



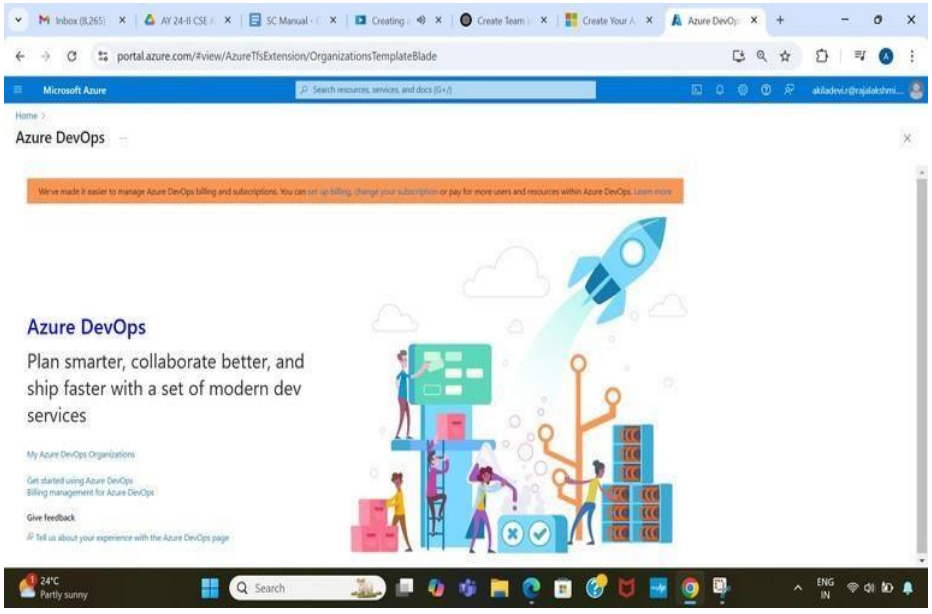
3. Azure home page

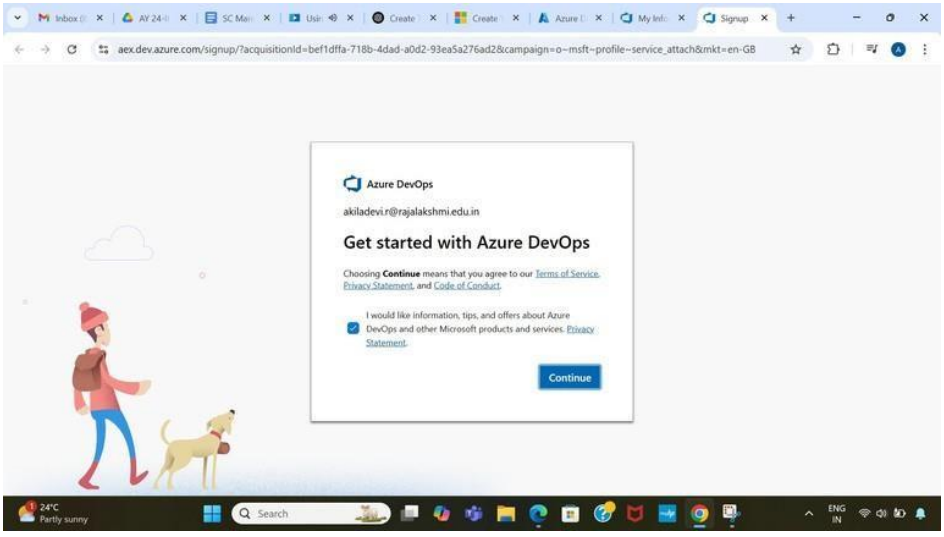


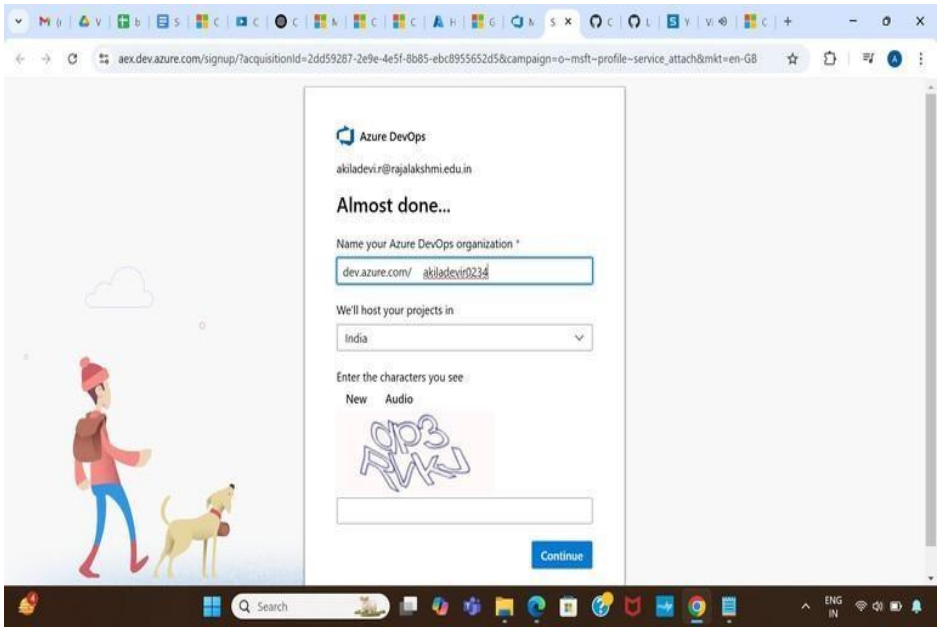
4. Open DevOps environment in the Azure platform by typing Azure DevOps Organizations in the search bar.



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.







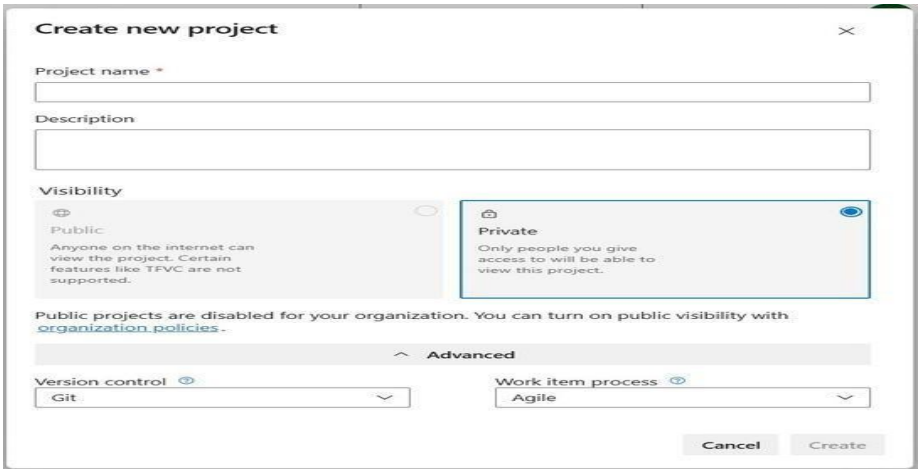
6. Create the First Project in Your Organization

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.

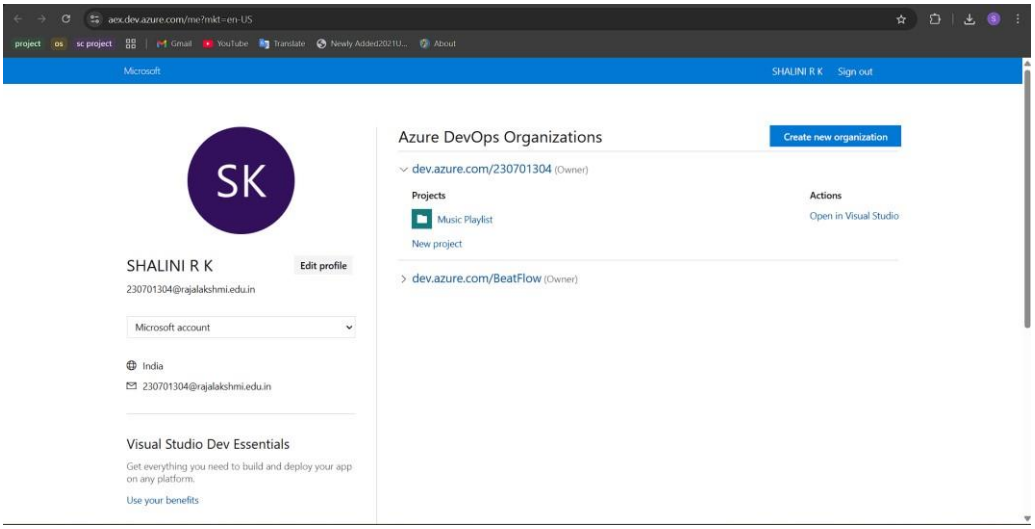
i. On the organization's **Home page**, click on the **New Project** button.
ii. Enter the project name, description, and visibility options:

- **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). Once

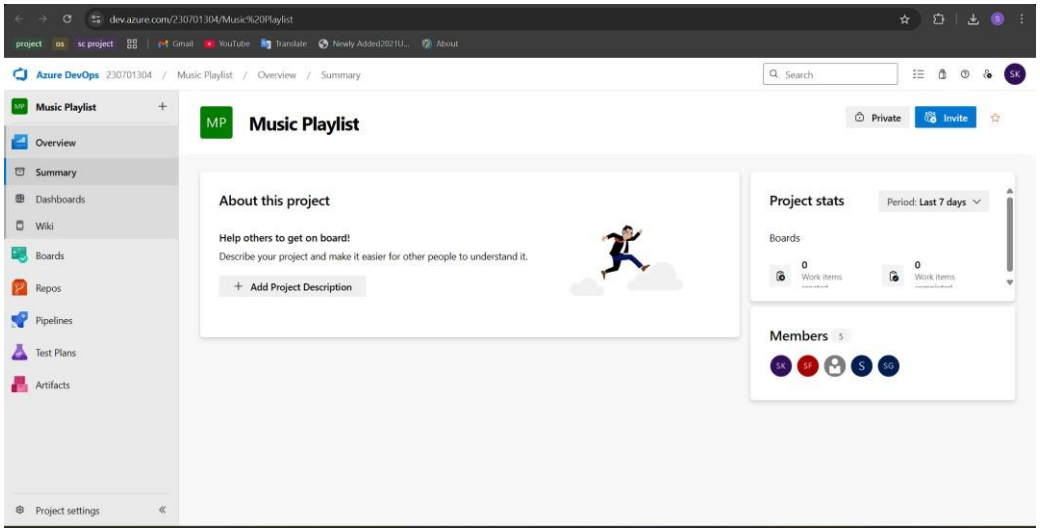
iii. you've filled out the details, click **Create** to set up your first project.



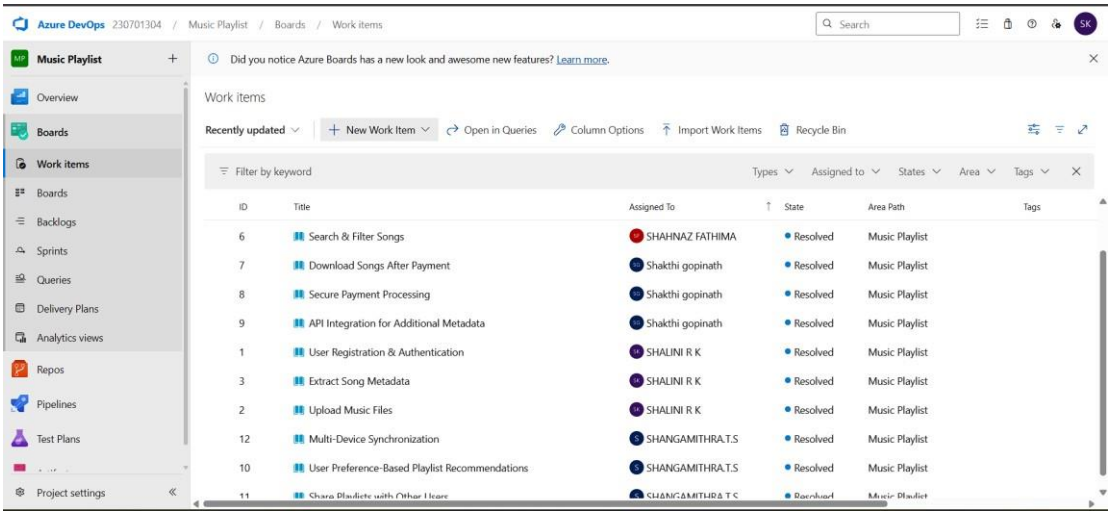
7. Once logged in, ensure you are in the correct organization. If you're part of multiple organizations, you can switch between them from the top left corner (next to your user profile). Click on the Organization name, and you should be taken to the Azure DevOps Organization Home page.



8. Project dashboard

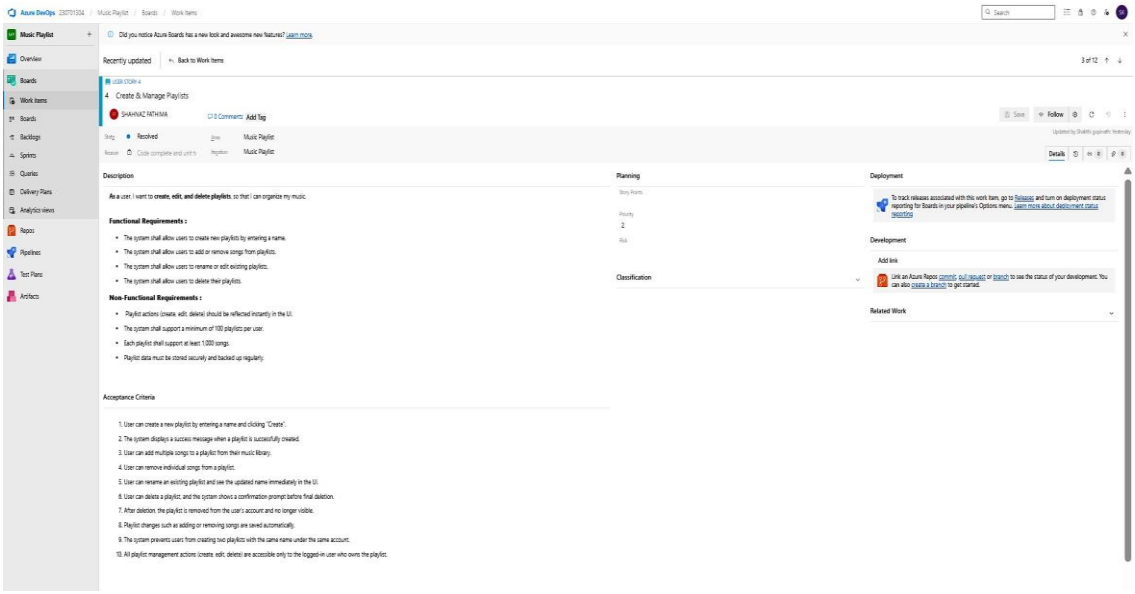


9. To manage user stories
   a. From the **left-hand navigation menu**, click on **Boards**. This will take you to the main **Boards** page, where you can manage work items, backlogs, and sprints.
   b. On the **work items** page, you'll see the option to **Add a work item** at the top. Alternatively, 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 User Story Details

11.



Result:

The user story was written successfully.

EXNO:5

SEQUENCE DIAGRAM

Aim:

To designa SequenceDiagramby usingMermaid.js

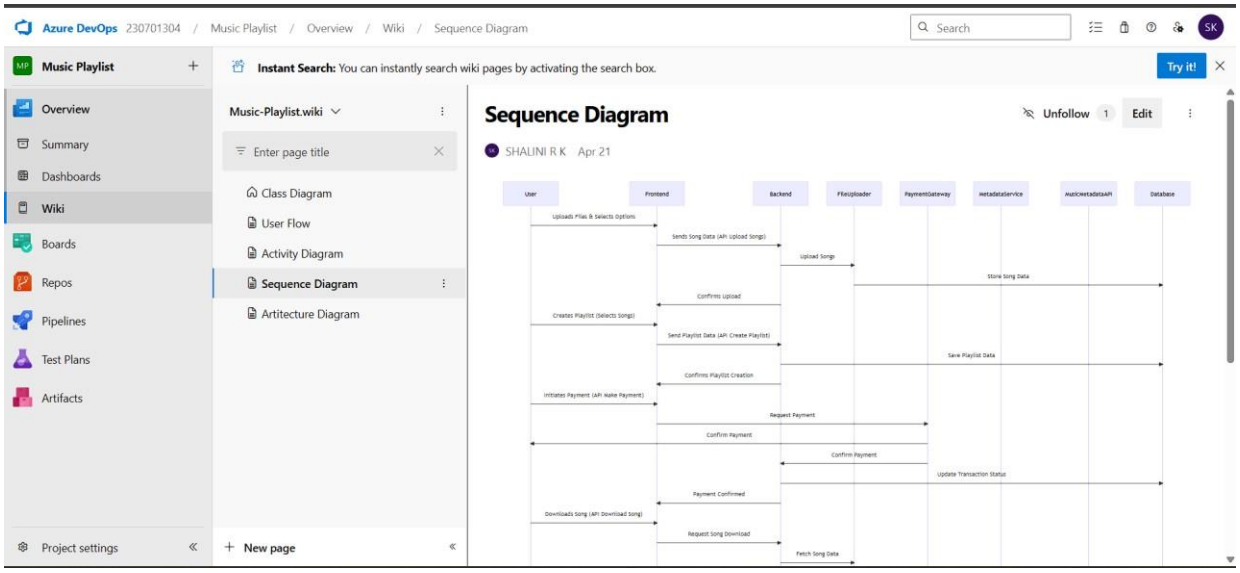
THEORY:

A Sequence Diagram is a key component of Unified Modelling 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 Opena projectin AzureDevOpsOrganisations.

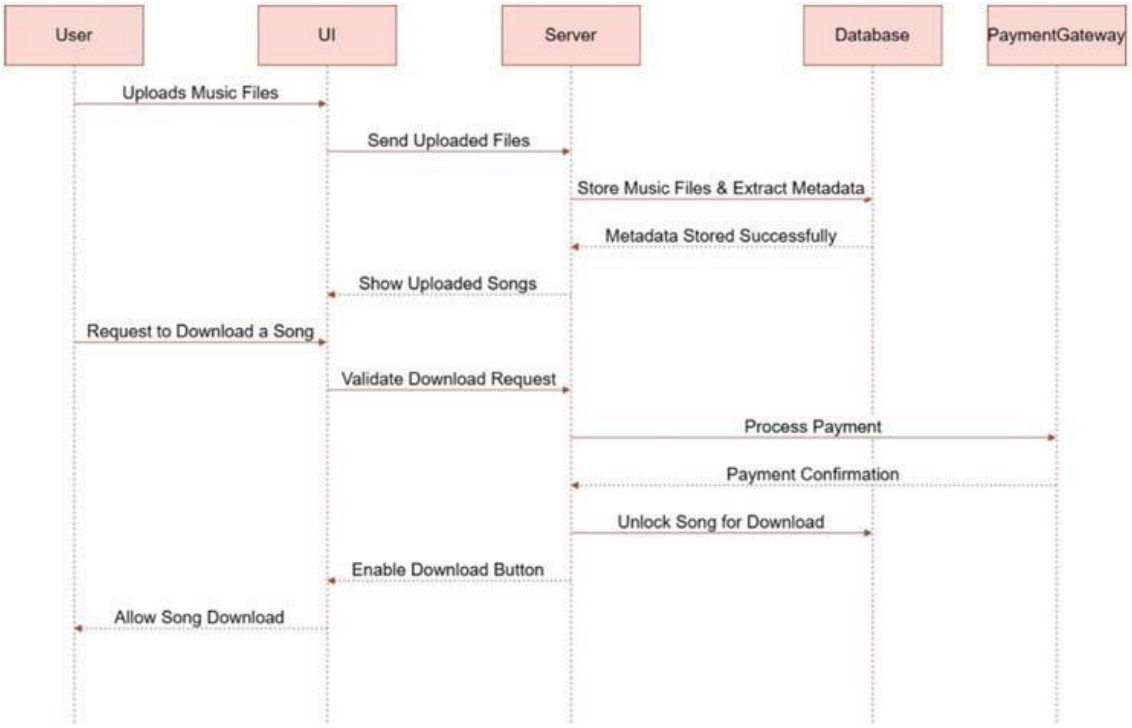
2 To designselectwiki frommenu



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

```
sequenceDiagram
    participant    User
    participant UI
    participant Server participant
    Database
    participant  PaymentGateway

    User->>UI: Uploads Music Files  UI-
    >>Server: Send Uploaded Files
    Server->>Database: Store Music Files & Extract Metadata
    Database-->>Server: Metadata Stored Successfully  Server--
    >>UI: Show Uploaded Songs
```

User->>UI: Request to Download a Song UI-
>>Server: Validate Download Request
Server->>PaymentGateway: Process Payment
PaymentGateway-->>Server: Payment Confirmation
Server->>Database: Unlock Song for Download Server-
>>UI: Enable Download Button
UI-->>User: Allow Song Download



Result:

The sequence diagram was drawn successfully.

EXNO.6

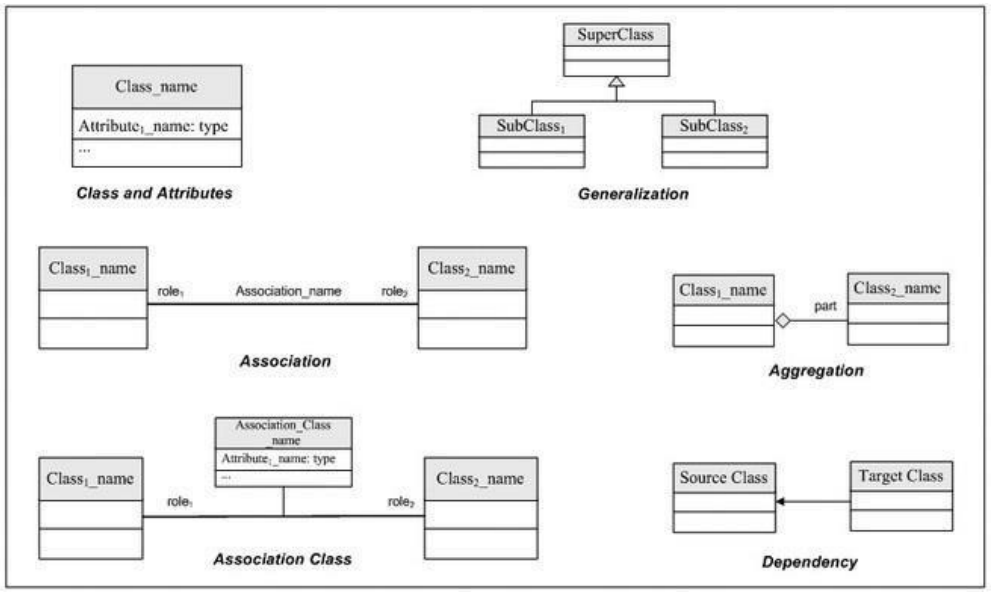
CLASSDIAGRAM

AIM :-

To draw a sampleclass diagramfor our projectMusic Playlist

THEORY

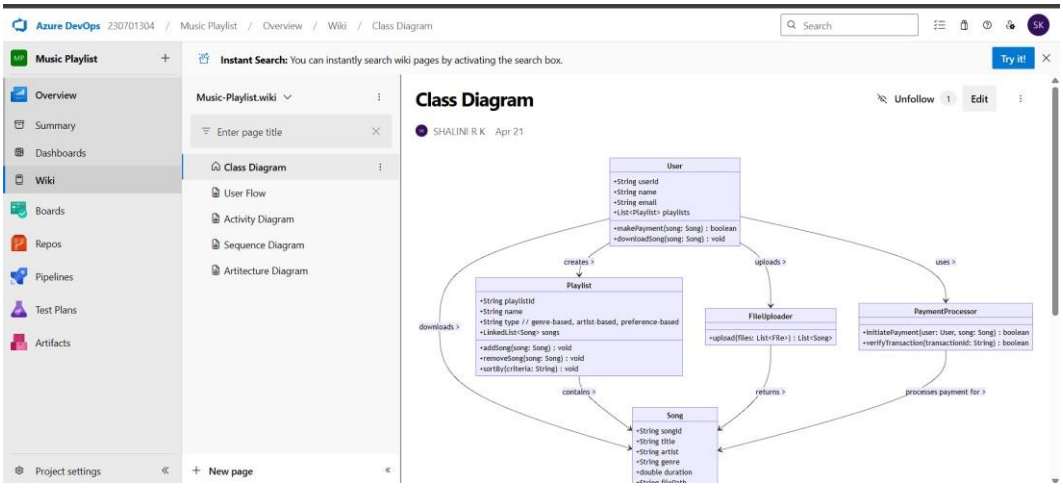
A UMLclass diagramis a visualtool that representsthe structureof a systemby showing its classes, attributes, methods, and the relationships between them.

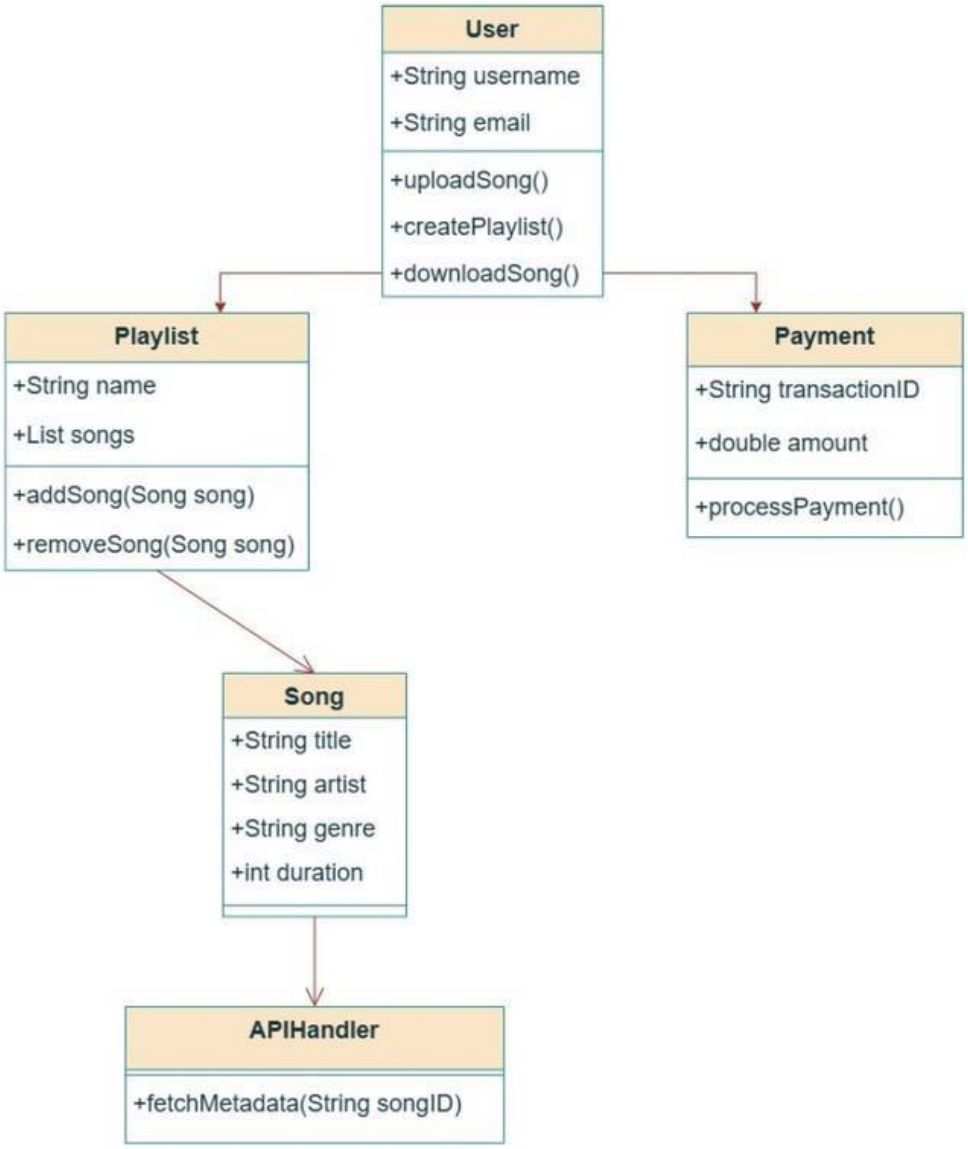


Notations in class diagram

Procedure:

1. Open a projectin AzureDevOpsOrganisations.
2. To designselect wiki from menu





Visit : https://mermaid.js.org/syntax/classDiagram.html

Result:

The use class diagram was designed successfully.

EX NO: 7

USECASE DIAGRAM

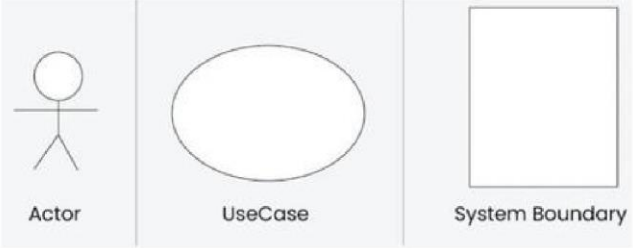
AIM:

Steps to draw the Use Case Diagram using draw.io

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

Step 1: Create the Use Case Diagram in Draw.io

- Open Draw.io (diagrams.net).
- Click "Create New Diagram" and select "Blank" or "UML Use Case" template.
- Add Actors (Users, Admins, External Systems) from the UML section.
- Add Use Cases (Functionalities) using ellipses.
- Connect Actors to Use Cases with lines (solid for direct interaction, dashed for <<include>> and <<extend>>).
- Save the diagram as .drawio or export as PNG/JPG/SVG.

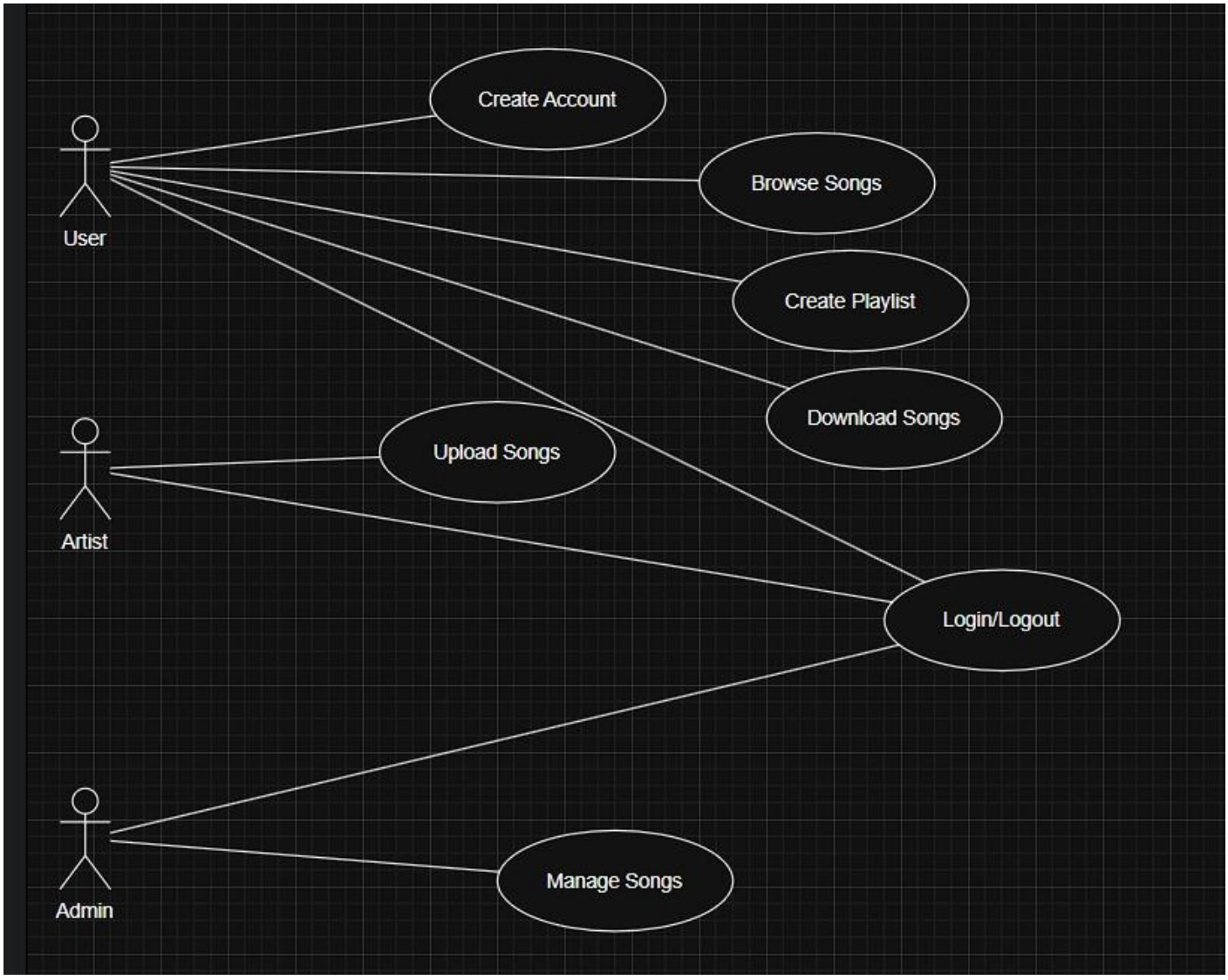
Step 2: Upload the Diagram to Azure DevOps Option

1: Add to Azure DevOps Wiki

- Open Azure DevOps and go to your project.
- Navigate to Wiki (Project > Wiki).
- Click "Edit Page" or create a new page.
- Drag & Drop the exported PNG/JPG image.
- Use Markdown to embed the diagram:
- ![Use Case Diagram](attachments/use_case_diagram.png)

Option 2: Attach to Work Items in Azure Boards

- Open Azure DevOps → Navigate to Boards (Project > Boards).
- Select a User Story, Task, or Feature.
- Click "Attachments" → Upload your Use Case Diagram.
- Add comments or descriptions to explain the use case.



Result:

The use case diagram was designed successfully

EXNO.8

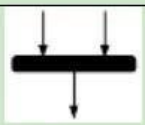
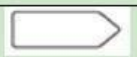
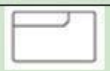
ACTIVITYDIAGRAM

AIM:-

To draw a sampleactivitydiagramfor your projector system.

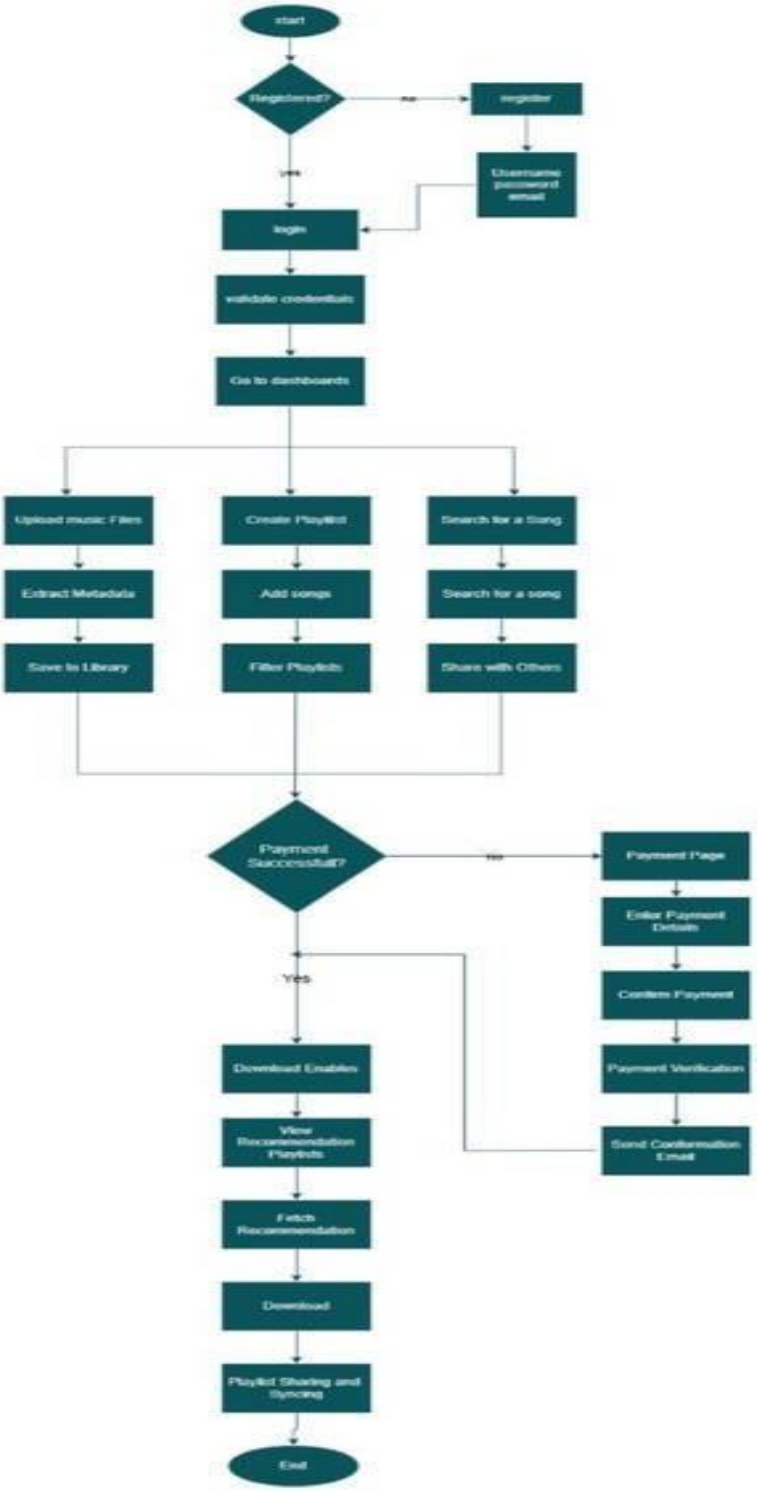
THEORY

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. Drawdiagramin draw.io
2. Uploadthe diagramin AzureDevOpswiki



Result:

The activity diagram was designed successfully

EX NO. 9

ARCHITECTURE DIAGRAM

AIM:

Steps to draw the Architecture Diagram using draw.io.

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.

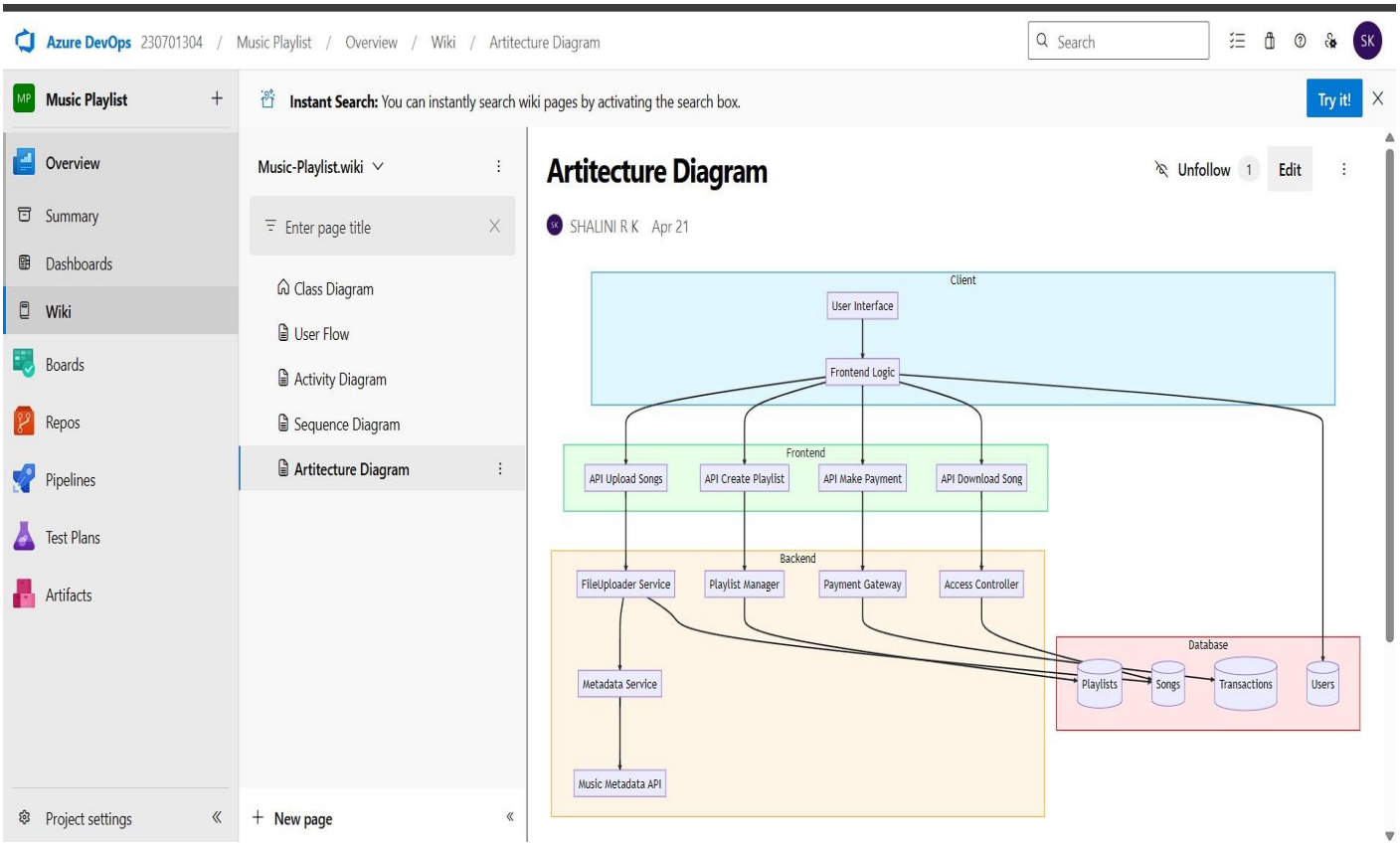
Procedure

1. Draw diagram in draw.io
2. Upload the diagram in Azure DevOps wiki

Code:

```
flowchart TD
%% Client Layer
subgraph Client
UI[User Interface]
UI --> FE[Frontend Logic]
end

%% Frontend Layer
subgraph Frontend
FE --> API1[API Upload Songs]
FE --> API2[API Create Playlist]
FE --> API3[API Make Payment]
FE --> API4[API Download Song]
End
```

```
%% Backend Layer
subgraph Backend
API1 --> FU[FileUploader Service]
API2 --> PM[Playlist Manager]
API3 --> PG[Payment Gateway]
API4 --> AC[Access Controller]
FU --> MS[Metadata Service]
MS --> EXT[Music Metadata API]
end

%% Database Layer
subgraph Database
DB1[(Users)]
DB2[(Songs)]
DB3[(Playlists)]
DB4[(Transactions)]
end

FU --> DB2
PM --> DB3
PG --> DB4
AC --> DB2
FE --> DB1

%% Optional Styling
style Client fill:#e0f7ff,stroke:#007acc,stroke-width:1px
style Frontend fill:#e6ffe6,stroke:#00cc66,stroke-width:1px
style Backend fill:#fff5e6,stroke:#ff9900,stroke-width:1px
style Database fill:#ffe6e6,stroke:#cc0000,stroke-width:1px
```



Result:

The architecture diagram was designed successfully.

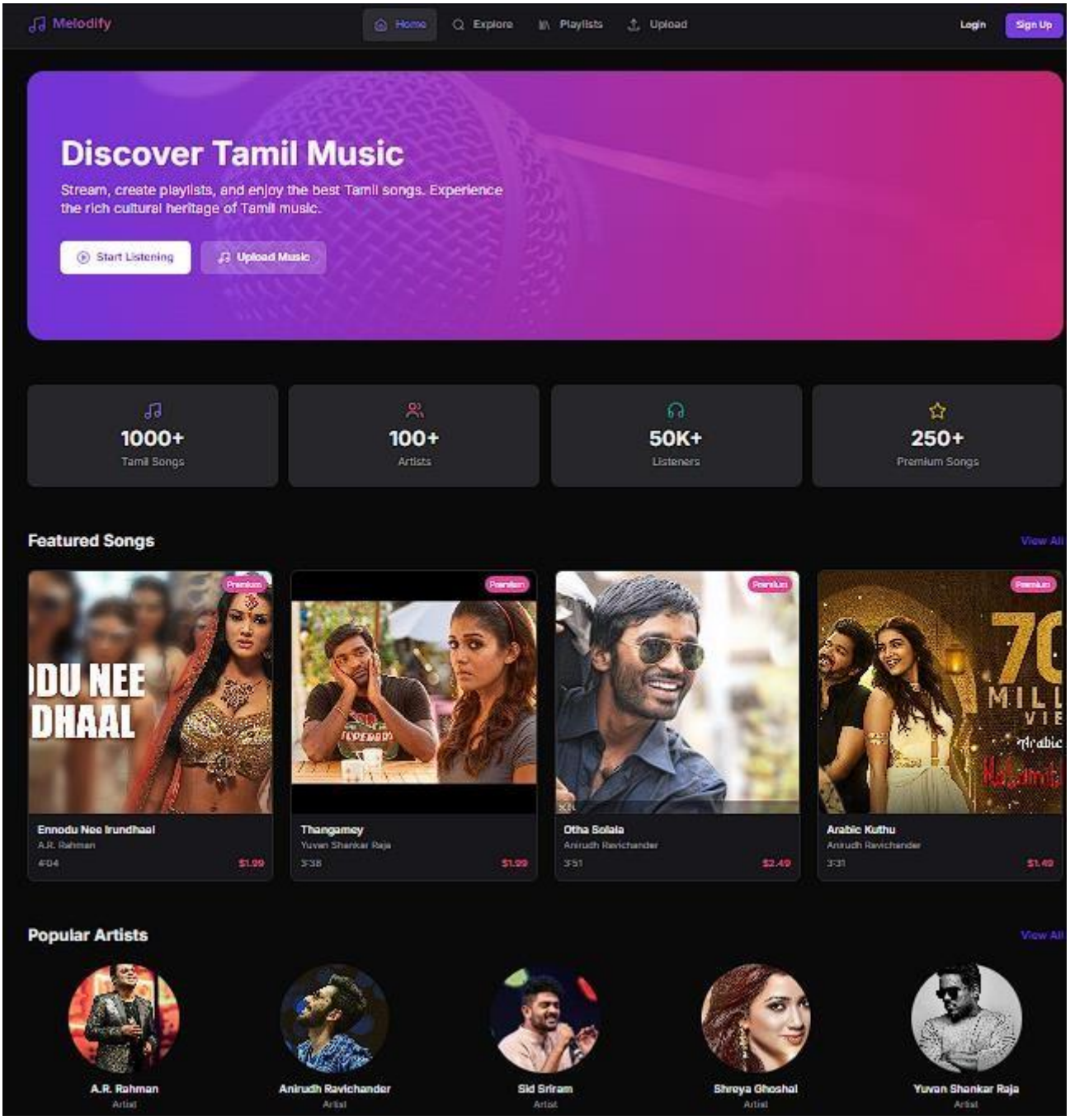
EX NO. 10

USER INTERFACE

AIM:

Design User Interface for the given project

Home Page:

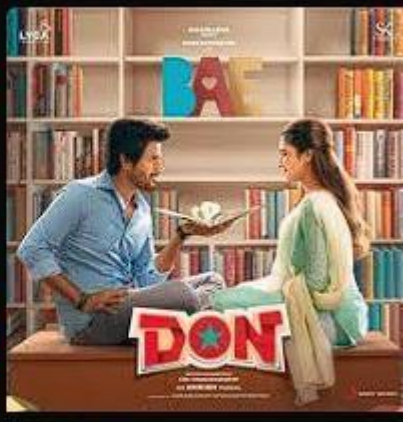


Featured Playlists

View A

Romantic Tamil Hits
A collection of the most romantic Tamil songs of all time

Tamil Dance Beats
Energetic Tamil songs that will make you dance



Melody Moments
Soft and melodious Tamil songs for your peaceful moments

Browse by Genre

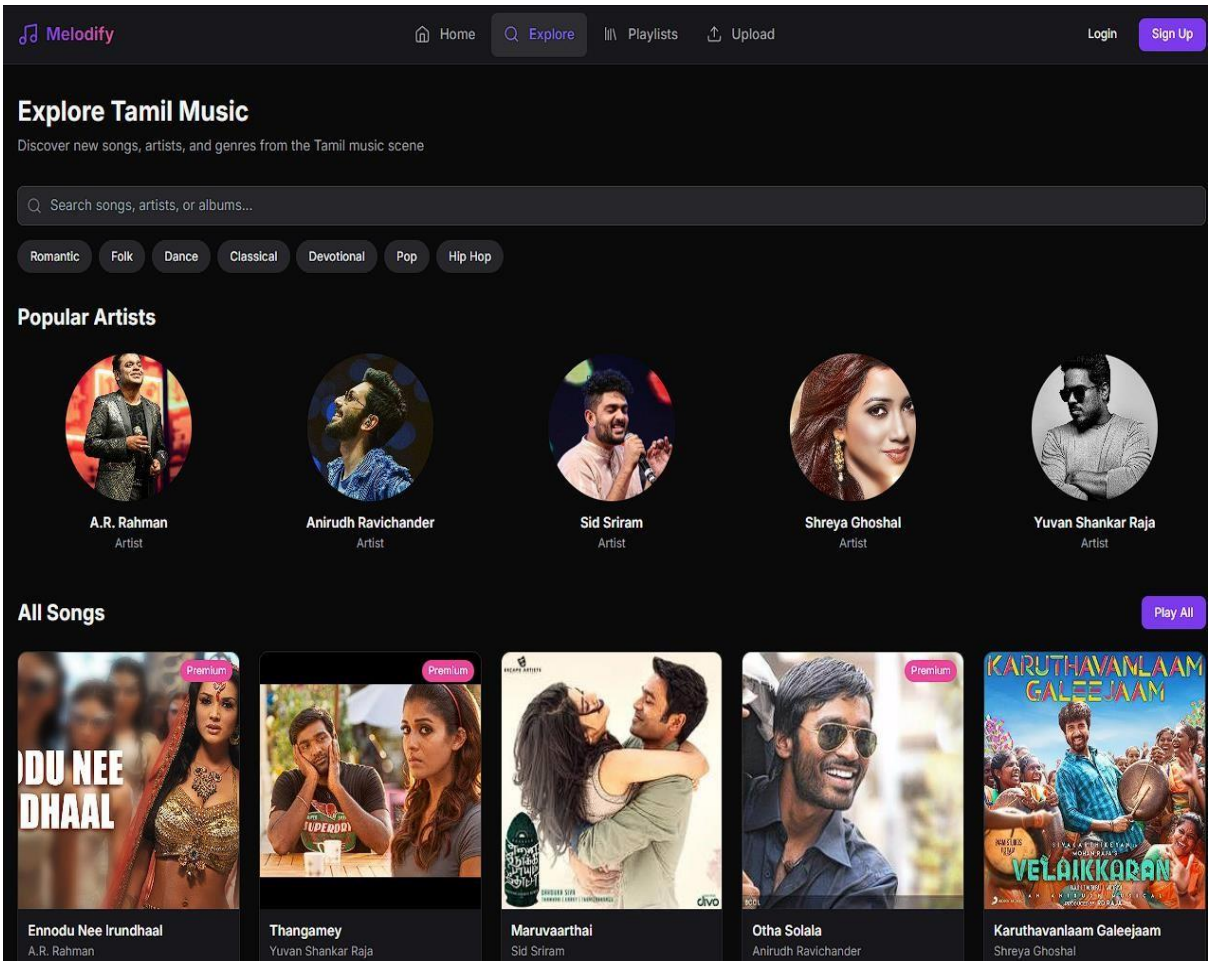
- Romantic
- Folk
- Dance
- Classical
- Devotional
- Pop
- Hip Hop

Ready to Enjoy Premium Tamil Music?

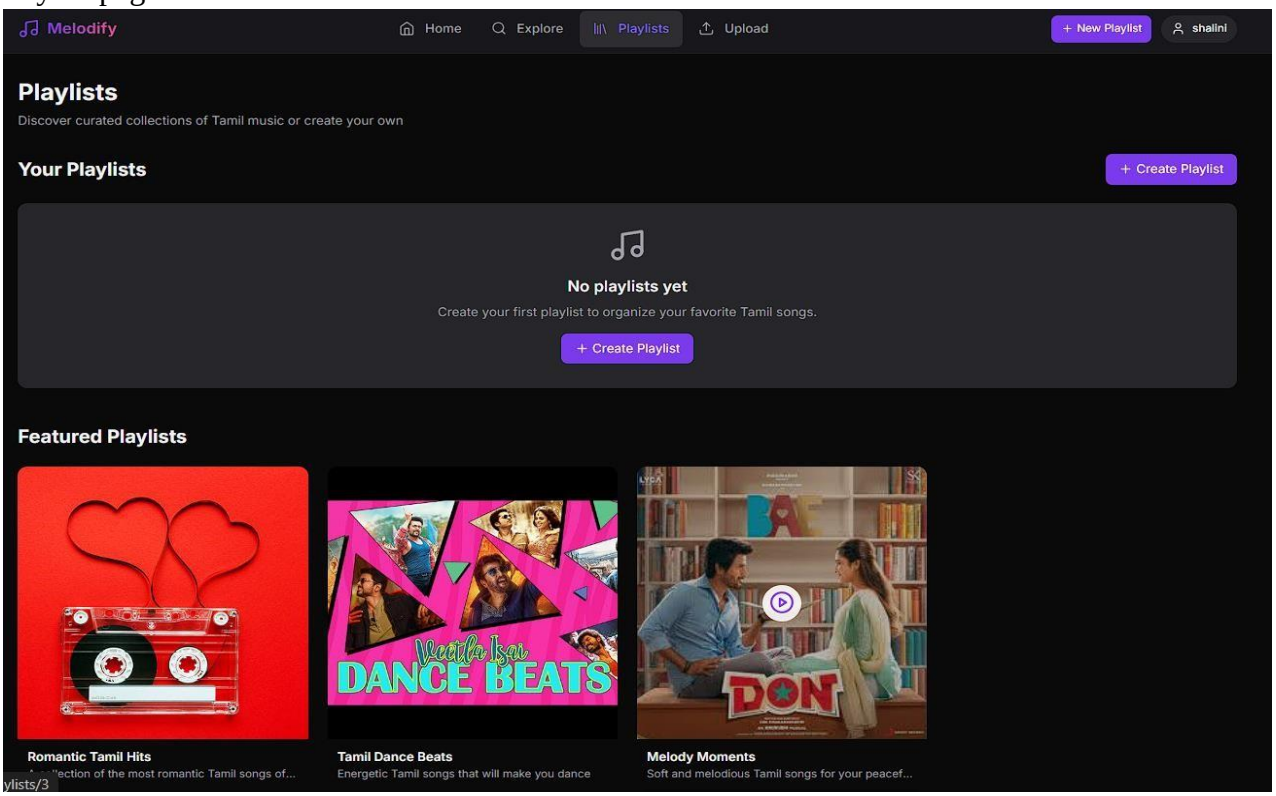
Create an account to unlock all features and enjoy premium songs.

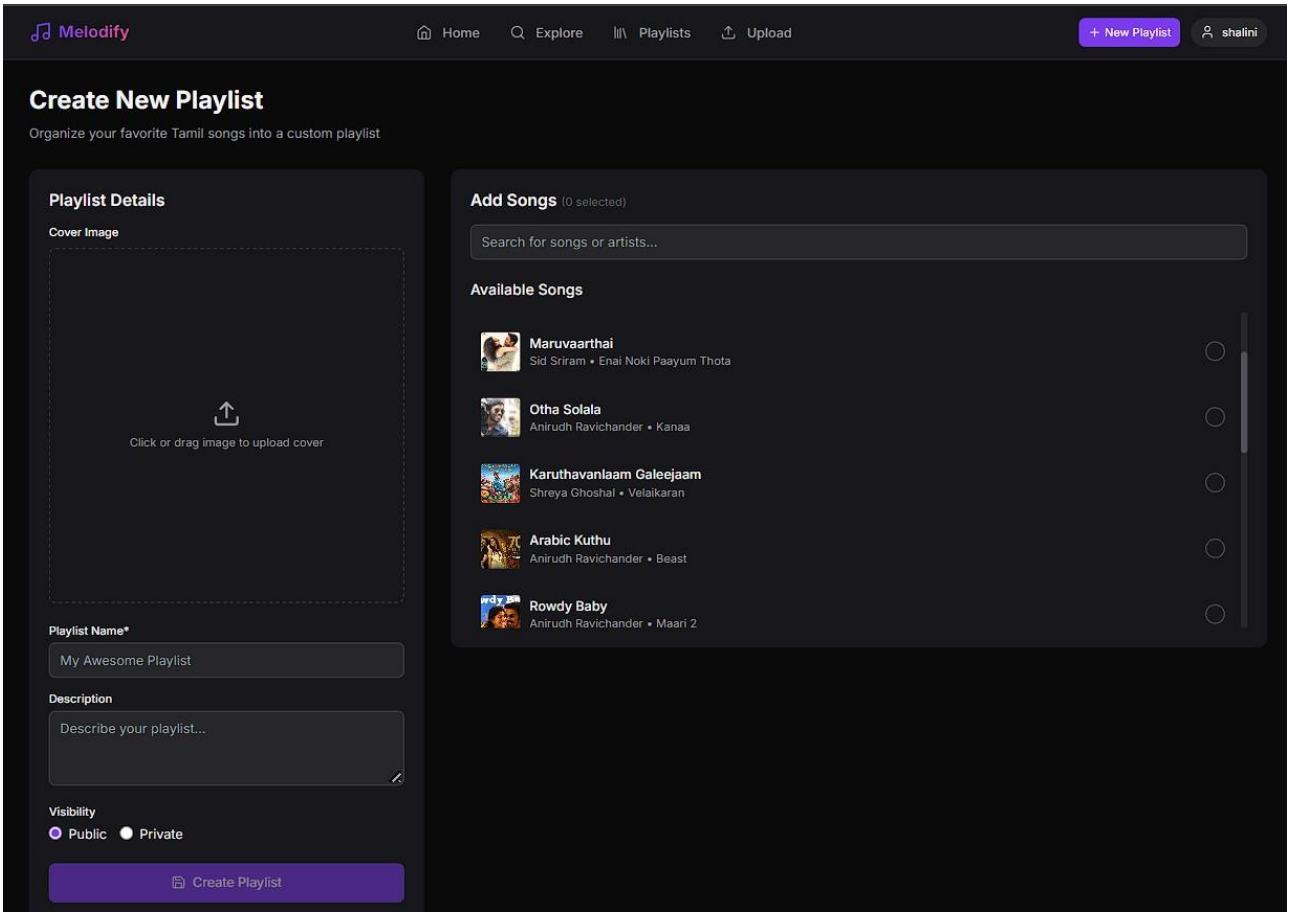
Sign Up Now

Explore Page:

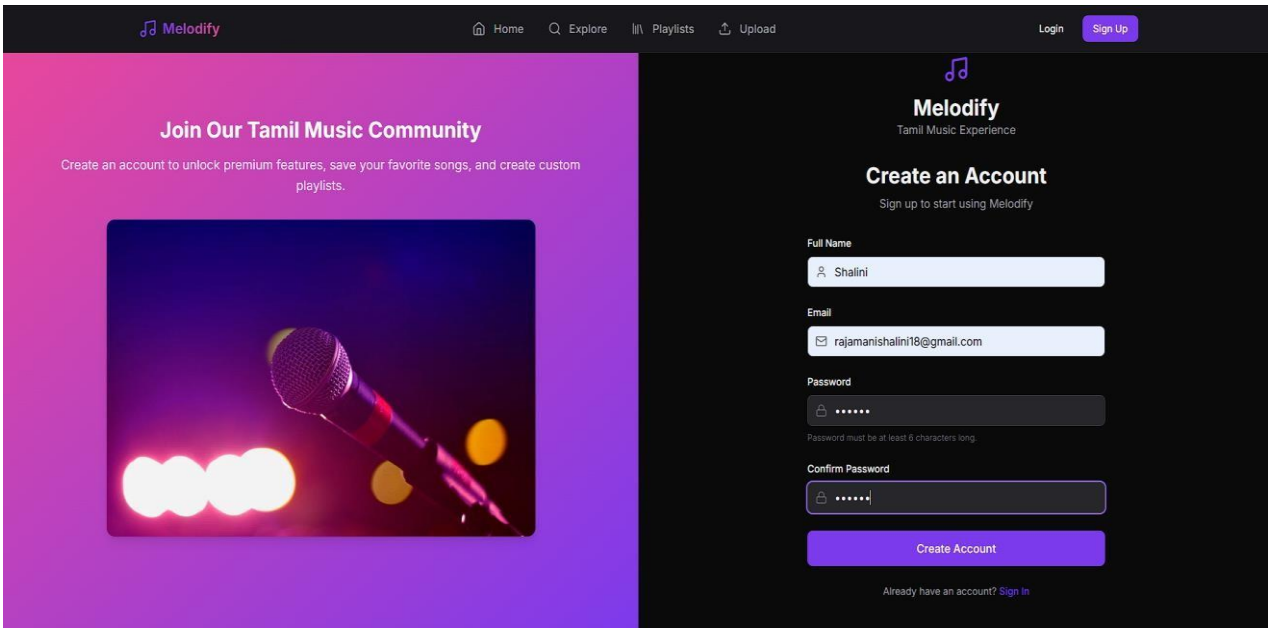


Playlist page:

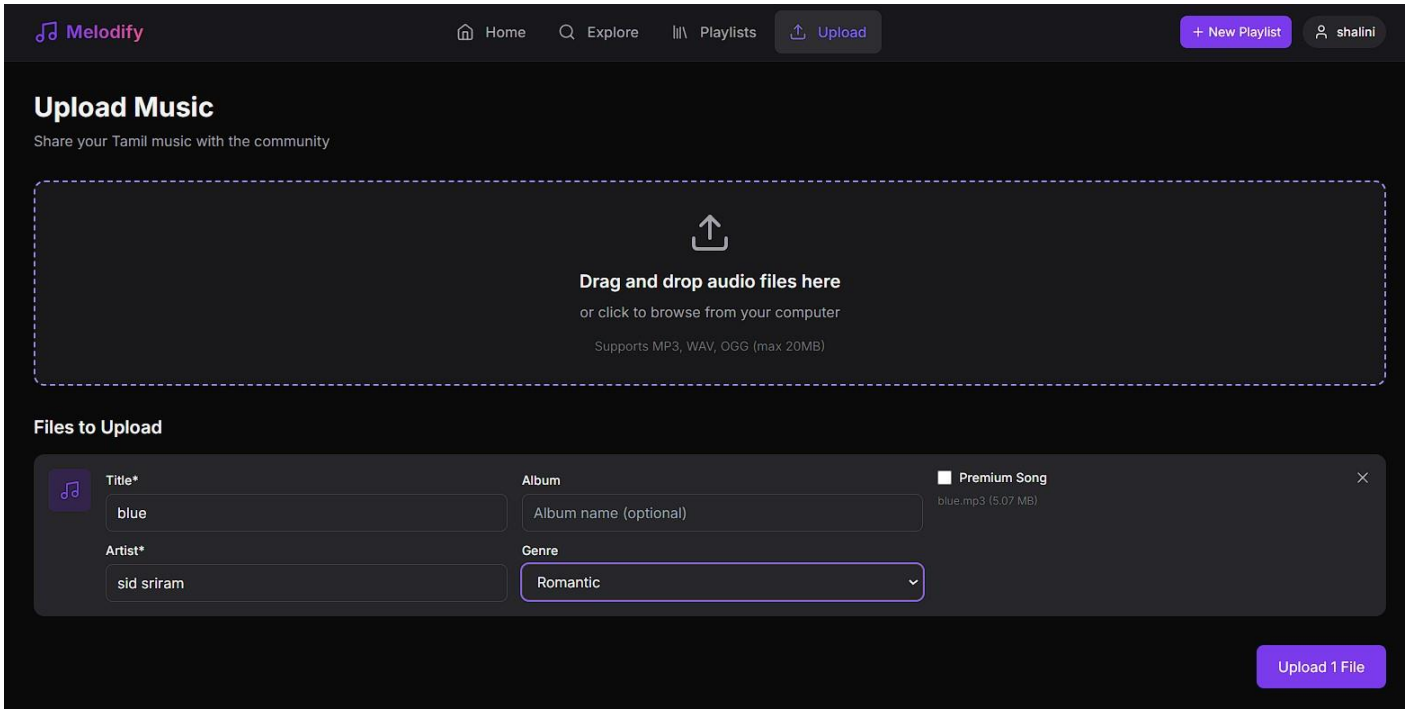




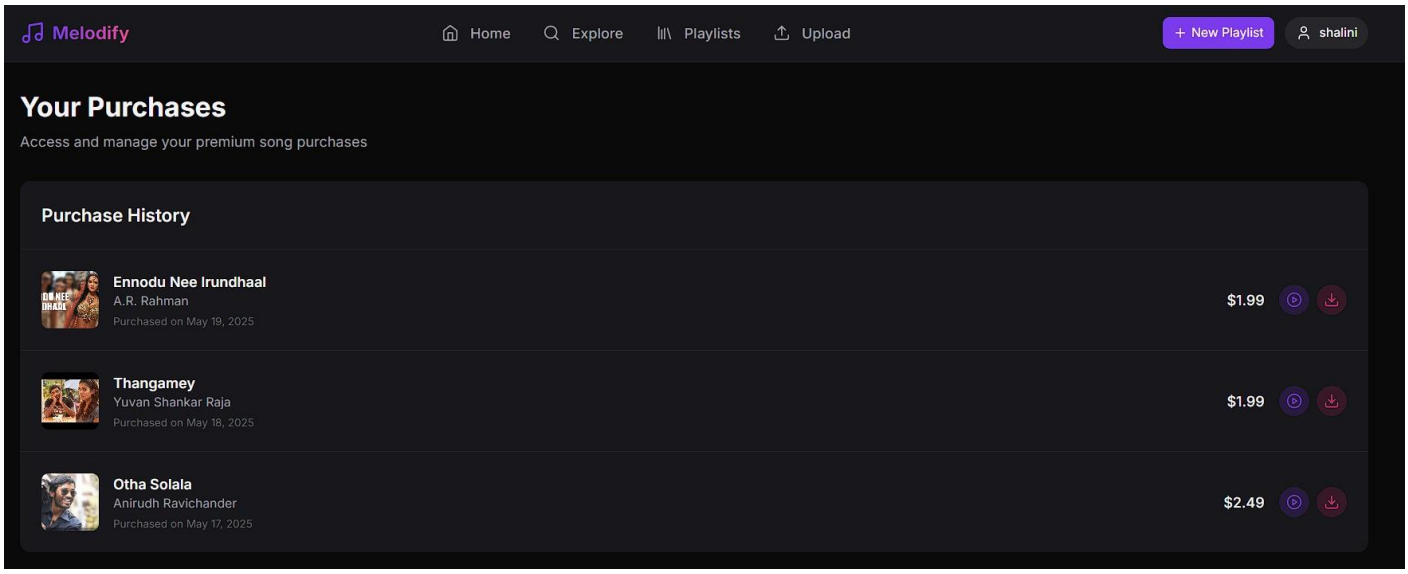
Sign In/Sign up Page:



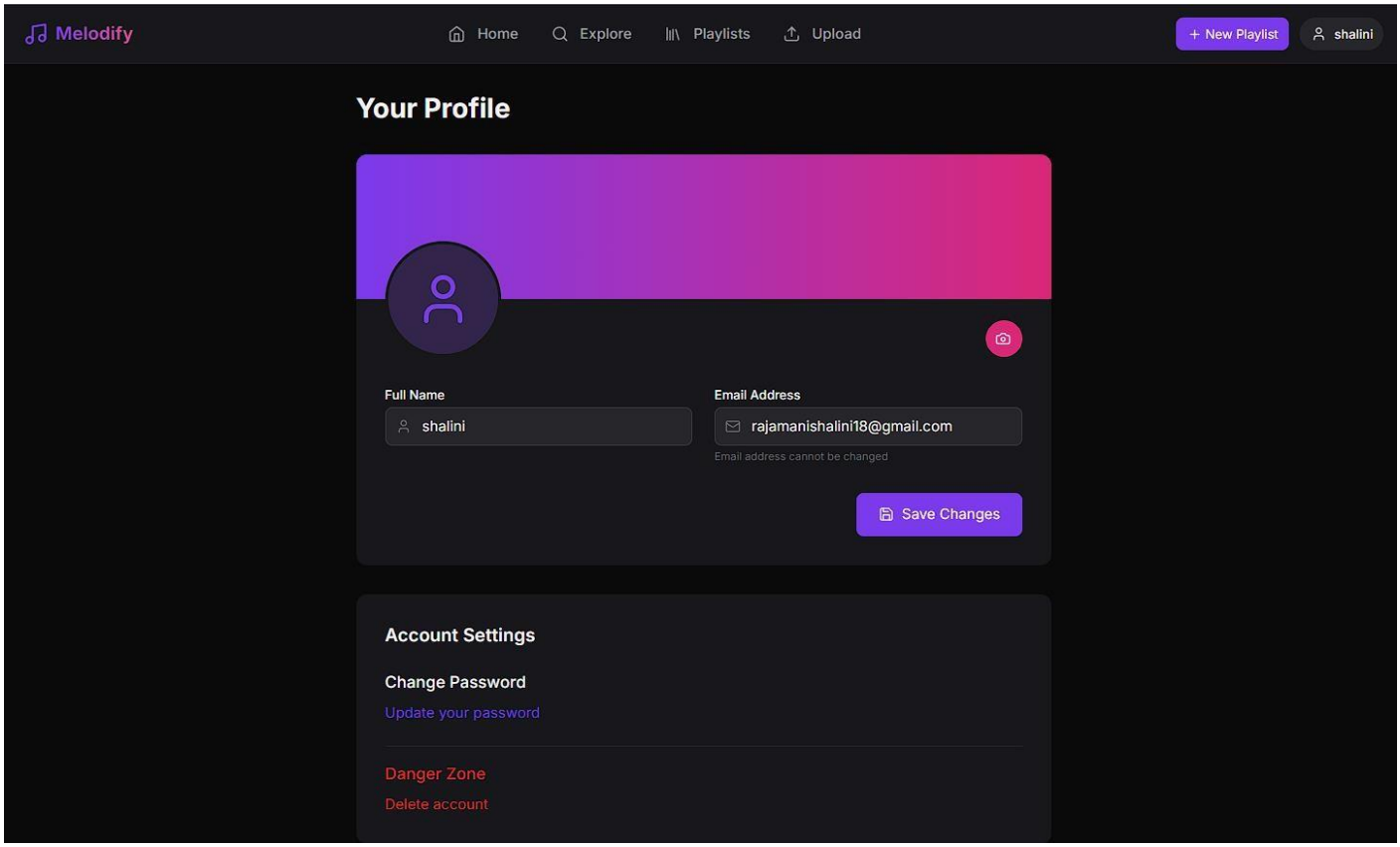
Upload Song Page:



Purchase Page:



Profile Page:



Result:

The UI was designedsuccessfully.

EX NO. 11

IMPLEMENTATION

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.

Result

Thus the application was successfully implemented.

EX NO. 12

CI-CD PIPELINE

AIM:

To implement the CI/CD pipeline in azure.

PROCEDURE:

Define CD Pipeline (Release Pipeline)

You can use either:

● Classic Releases (GUI-based) ●
YAML-based Continuous Delivery

GUI-based Example:

1. Go to Pipelines > Releases
2. Create a new release pipeline
3. Artifact: Link to the CI build artifact
4. Stage: Add tasks for Azure App Service, VM, or AKS deployment

Task Example (App Service Deploy):

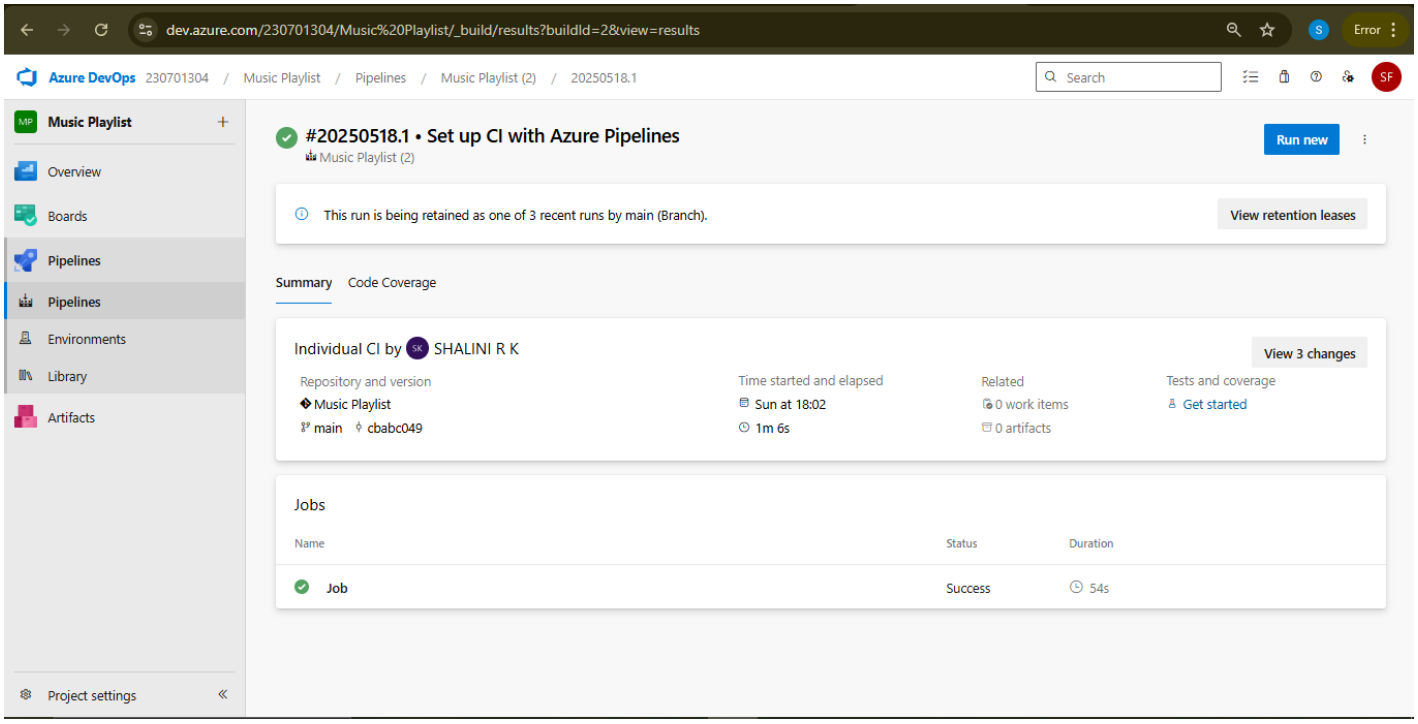
```
- task: AzureWebApp@1
inputs:
  azureSubscription: '<your-azure-
connection>'    appType: 'webApp'
appName: '<your-app-name>'
  package: '$(System.DefaultWorkingDirectory)/**/*.zip'
```

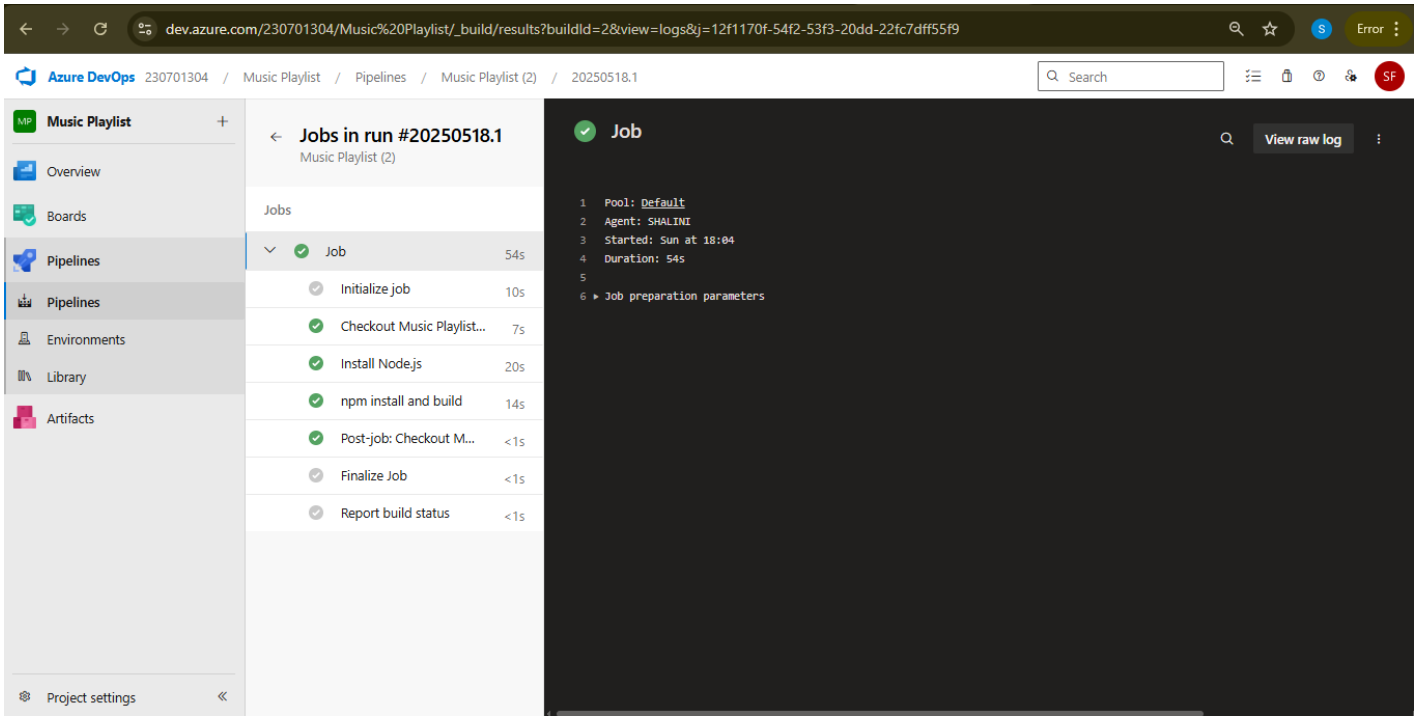
Connect Azure Subscription

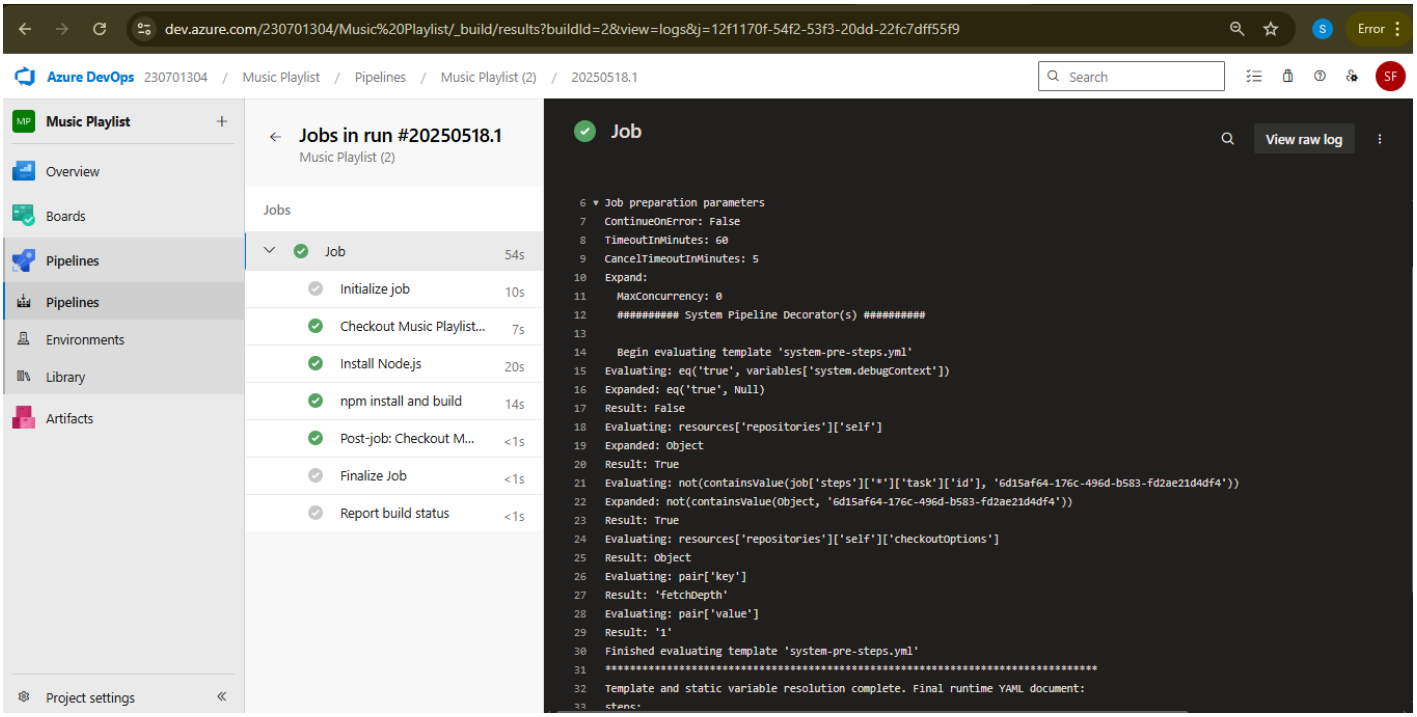
- From Project Settings > Service Connections
- Add a new Azure Resource Manager (ARM) connection

Enable Continuous Deployment Trigger

● For Release Pipelines: Enable CD trigger to auto-deploy on new build artifacts.


dev.azure.com/230701304/Music%20Playlist/_build/results?buildId=2&view=results
Azure DevOps 230701304 / Music Playlist / Pipelines / Music Playlist (2) / 20250518.1
Search
Music Playlist
Overview
Boards
Pipelines
Pipelines
Environments
Library
Artifacts
Project settings
#20250518.1 • Set up CI with Azure Pipelines
Music Playlist (2)
Run new
This run is being retained as one of 3 recent runs by main (Branch).
View retention leases
Summary
Code Coverage
Individual CI by SHALINI R K
View 3 changes
Repository and version
Music Playlist
main
cbabc049
Time started and elapsed
Sun at 18:02
1m 6s
Related
0 work items
0 artifacts
Tests and coverage
Get started
Jobs
Name
Status
Duration
Job
Success
54s


dev.azure.com/230701304/Music%20Playlist/_build/results?buildId=2&view=logs&j=12f1170f-54f2-53f3-20dd-22fc7dff55f9
Azure DevOps 230701304 / Music Playlist / Pipelines / Music Playlist (2) / 20250518.1
Search
Music Playlist
Overview
Boards
Pipelines
Pipelines
Environments
Library
Artifacts
Project settings
Jobs in run #20250518.1
Music Playlist (2)
Jobs
Job 54s
Initialize job 10s
Checkout Music Playlist... 7s
Install Node.js 20s
npm install and build 14s
Post-job: Checkout M... <1s
Finalize Job <1s
Report build status <1s
Job
View raw log
1 Pool: Default
2 Agent: SHALINI
3 Started: Sun at 18:04
4 Duration: 54s
5
6 ▸ Job preparation parameters



RESULT:

Thus the CI/CD pipeline was successfully implemented.