SOFTWARE ENGINEERING & CONCEPTS – LAB MANUAL

Grade Management System

Tharun.R.L , 220701302

BE CSE / 2nd Year / E section

Software Concepts & Engineering - Lab Manual

STUDENT GRADE MANAGEMENT SYSTEM

Team 2

R.L Tharun (220701302)
S.J Vanisree (220701308)
V. Chandra Harsha (220701314)
S.T Vijayetha (220701320)
Yashwanth Ramesh (220701326)
G. Karthik (220701504)

Software Concepts & Engineering - Lab Manual

Software Concepts & Engineering - Lab Manual

Overview of the Project

Project Description
The Student Grade Management System (SGMS) is designed to streamline the process of recording, managing, and analyzing student grades. It aims to provide an efficient, user-friendly platform for teachers, administrators, students, and parents to access and manage academic performance data.
Problem Statement
The SGMS seeks to resolve several issues prevalent in traditional grading systems, which are often manual, inefficient, and prone to errors. Key problems include:

1. **Inefficiency and Time Consumption**: Manually entering and calculating grades is time-consuming for educators.
2. **Human Error**: Manual processes are susceptible to errors in data entry and calculation.
3. **Lack of Real-Time Access**: Students and parents often have limited access to up-to-date academic performance data.
4. **Data Management**: Storing and retrieving student records can be cumbersome, especially in large institutions.
5. **Analysis and Reporting**: Generating reports and analyzing academic performance trends is difficult without automated tools.

Data Management Perspective
The SGMS addresses these problems by efficiently managing and utilizing data:

1. **Data Collection**:
    - **Student Information**: Personal details, enrollment information, and course registrations.
    - **Grades**: Scores from assignments, quizzes, exams, and other assessments.
    - **Attendance**: Records of student attendance.
2. **Data Storage**:
    - Centralized database to securely store student information and grades.
    - Use of relational databases (e.g., MySQL, PostgreSQL) to ensure data integrity and facilitate complex queries.
3. **Data Processing**:
    - Automated calculations of final grades based on predefined criteria (e.g., weighted averages).
    - Validation checks to ensure data accuracy during entry.
4. **Data Analysis and Reporting**:
    - Generation of detailed reports on student performance.

- Statistical analysis to identify trends, such as class averages and student progress over time.
- Tools for predictive analysis to foresee potential academic issues.

Benefits of Implementing the System

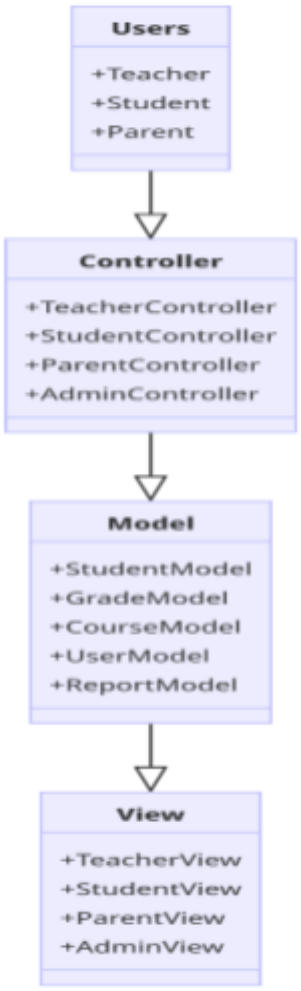
1. **Efficiency**:
   - Reduces the time educators spend on administrative tasks.
   - Automates repetitive processes like grade calculations and report generation.
2. **Accuracy**:
   - Minimizes errors in grade entry and calculations.
   - Ensures data consistency through validation mechanisms.
3. **Accessibility**:
   - Provides real-time access to grades and performance reports for students and parents through an online portal.
   - Facilitates communication between teachers, students, and parents.
4. **Data Management**:
   - Simplifies the storage, retrieval, and management of academic records.
   - Scales efficiently to accommodate a large number of students and courses.
5. **Enhanced Analysis**:
   - Offers tools for comprehensive analysis and reporting of academic performance.
   - Helps in identifying areas where students may need additional support or intervention.
6. **Transparency and Engagement**:
   - Increases transparency in grading processes.
   - Encourages student engagement by providing clear insights into their academic performance.

By addressing these critical areas, the SGMS not only improves operational efficiency but also enhances the educational experience for all stakeholders involved. The system's robust data management capabilities ensure that academic records are accurate, accessible, and useful for decision-making and strategic planning.

-

Business Architecture Diagram

MVC Architecture



Manual Process

1. Data Entry:
    - Teachers: Enter student grades manually into paper gradebooks or simple spreadsheets.
    - Administrators: Collect these grade records at the end of the term for archival.
2. Grade Calculation:
    - Teachers: Manually calculate final grades using calculators or spreadsheet formulas. Verify the accuracy of these calculations.
3. Record Keeping:
    - Teachers: Maintain individual gradebooks.

- Administrators: Store paper records in filing cabinets or centralized digital records in isolated systems.

4. Report Generation:
   - Teachers: Manually compile grades into reports.
   - Administrators: Collect these reports for distribution.
5. Communication:
   - Teachers: Communicate grades to students and parents during class or parent-teacher meetings.
   - Administrators: Handle grade disputes and requests for official transcripts.

Automatic Process (If Any)

1. Data Entry:
   - Teachers: Enter grades into basic school management software.
   - Administrators: Monitor data entry and maintain the software.
2. Grade Calculation:
   - Teachers: Use the software's built-in functions to calculate final grades.
   - Administrators: Ensure the software is configured correctly for accurate calculations.
3. Record Keeping:
   - Teachers: Input data into the system, which saves it to a centralized database.
   - Administrators: Manage the database, ensuring data integrity and backup.
4. Report Generation:
   - Teachers: Generate reports through the software, which automates the process.
   - Administrators: Oversee the report generation and distribution process.
5. Communication:
   - Teachers: Share grades with students and parents through the software, which might provide limited online access.
   - Administrators: Handle advanced communication needs and software management.

Personas and Current Process

1. Teachers:
   - Current Process: Enter grades manually or into basic software, calculate final scores, and generate reports.
   - Challenges: Time-consuming, prone to errors, and difficult to manage multiple classes.
2. Administrators:
   - Current Process: Collect grades from teachers, verify accuracy, store records, and handle report distribution.
   - Challenges: Managing large volumes of data, ensuring data accuracy, and handling data retrieval requests.
3. Students:

- Current Process: Receive grades periodically through physical reports or limited online systems.
- Challenges: Delayed feedback, limited access to performance data, and difficulty in tracking academic progress.

4. Parents:
    - Current Process: Obtain grade information during scheduled meetings or through periodic reports.
    - Challenges: Limited visibility into their child's performance, delayed access to important academic information.

Business Problems

1. Efficiency:
    - Problem: Manual processes are slow and labor-intensive.
    - Impact: Teachers and administrators spend excessive time on data entry and calculations, reducing their availability for other important tasks.
2. Accuracy:
    - Problem: High potential for human error in manual data entry and calculations.
    - Impact: Inaccurate grades can negatively affect students' academic records and lead to disputes.
3. Accessibility:
    - Problem: Students and parents have limited and delayed access to grades.
    - Impact: Students may not receive timely feedback to improve their performance, and parents cannot adequately support their children's academic needs.
4. Data Management:
    - Problem: Fragmented and unorganized record-keeping.
    - Impact: Difficulties in retrieving and maintaining accurate academic records, leading to potential data loss and inefficiency in handling requests.
5. Communication:
    - Problem: Inefficient communication channels between teachers, students, and parents.
    - Impact: Lack of timely updates and feedback, hindering effective academic support and intervention.
6. Analytical Capability:
    - Problem: Lack of tools for comprehensive analysis and reporting of academic performance.
    - Impact: Missed opportunities for early intervention, performance tracking, and strategic planning.

Requirements as User Stories

User Story 1 : As a user, I want to be able to log in securely to access the system so that I can protect my account information.

Acceptance Criteria :

- Users can log in using their email and password.
- Multi-factor authentication is optional but available for users who want to enable it.
- Session tokens expire after a period of inactivity to prevent unauthorized access.

User Story 2 : As an admin, I want to control user access levels and permissions so that I can manage who can view and modify data.

Acceptance Criteria :

- Admins can assign roles and permissions to users.
- Access to certain features and data is restricted based on user roles.
- Audit logs record all user actions, including login attempts, role changes, and data modifications.

User Story 3 : As an admin, I want to be able to manage student records so that I can keep the database up to date.

Acceptance Criteria :

- Admins can add new student records with required information such as name, email, and student ID.
- Data validation prevents the addition of duplicate student records and ensures data consistency.

User Story 4 : As an admin, I want to be able to search for specific student records quickly so that I can retrieve information efficiently.

Acceptance Criteria :

- Admins can search for student records by name, email, or student ID.
- Search results are displayed in real-time as the user types.
- Autocomplete suggestions provide accurate matches for student names and IDs.

User Story 5 : As a faculty member, I want to input marks for assigned subjects so that I can accurately record student performance.

Acceptance Criteria :

- Faculty members can input marks for each assigned subject and student.
- Marks are validated to ensure they fall within the specified range (e.g., 0-100).
- Successfully entered marks are saved and reflected in the student's record immediately.

User Story 6 : As a faculty member, I want to receive feedback on the validity of the entered marks so that I can correct any errors promptly.

Acceptance Criteria :

- Invalid mark entries trigger error messages indicating the reason for the rejection (e.g., out of range, non-numeric).
- Error messages provide clear instructions on how to correct the issue.
- Error logs capture details of failed mark submissions for review by administrators.

User Story 7 : As a student, I want to be able to view my marks for each subject so that I can track my academic progress.

Acceptance Criteria :

- Students can view their marks for each subject in a clear and organized manner.
- Marks are sorted by subject and displayed with corresponding grades (if applicable).
- The dashboard loads quickly, even with a large number of marks.

User Story 8 : As a student, I want to see my overall grade so that I can understand my performance at a glance.

Acceptance Criteria :

- The overall grade is calculated accurately based on the weighted average of subject marks.
- The overall grade is prominently displayed on the student dashboard and updated in real-time.
- Changes in subject marks trigger automatic updates to the overall grade.

User Story 9 : As a student, I want to see graphical representations of my performance over time so that I can identify trends and patterns.

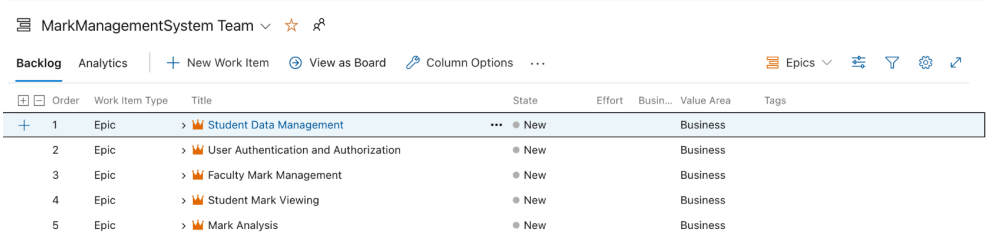
Acceptance Criteria :

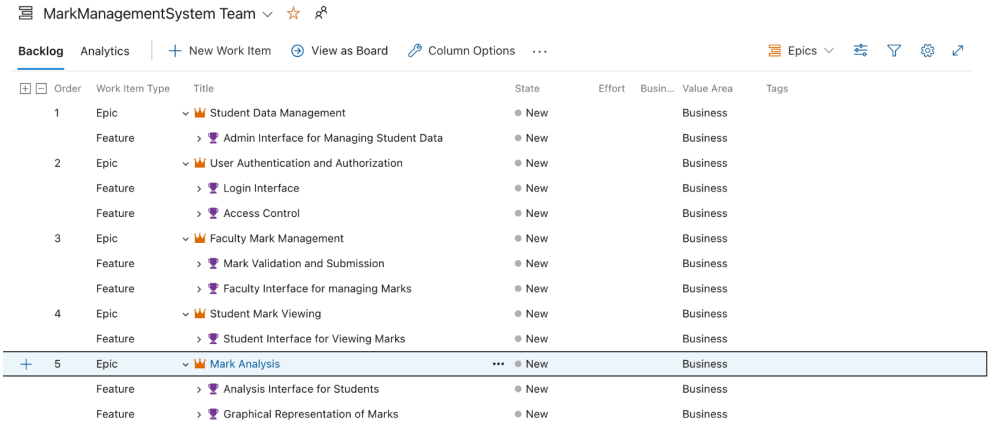
- Graphs display student performance trends across semesters for selected subjects.
- Users can interact with graphs to zoom in on specific time periods and subjects.
- Performance data is represented accurately, with clear labels and legends for easy interpretation.

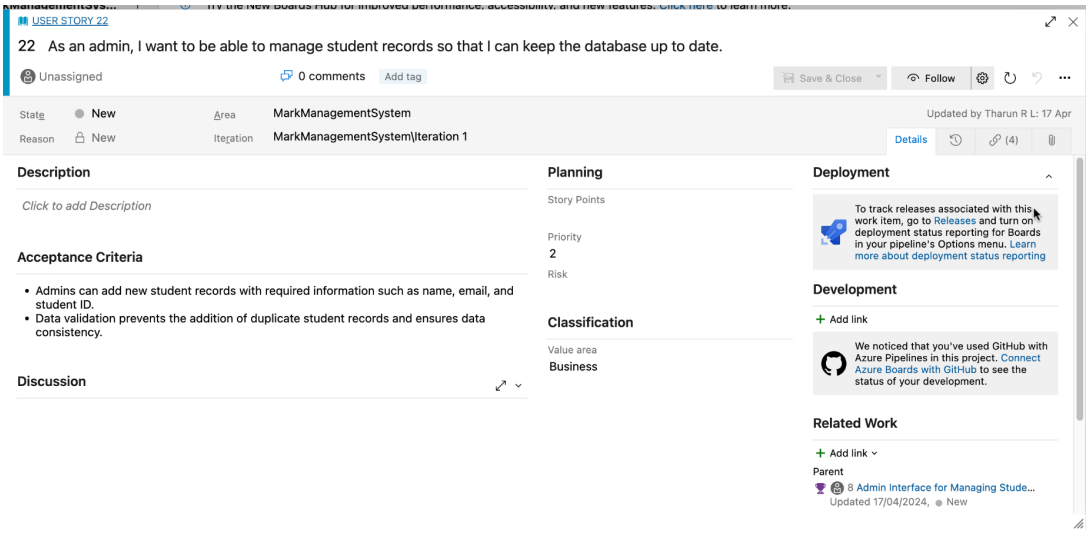
User Story 10 : As a student, I want to compare my performance with the class average so that I can gauge my standing relative to my peers.

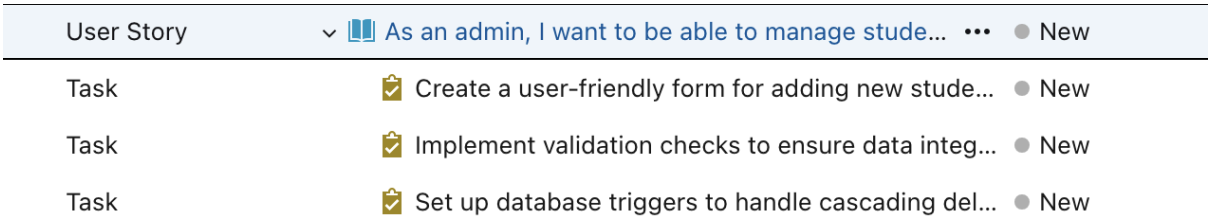
Acceptance Criteria :

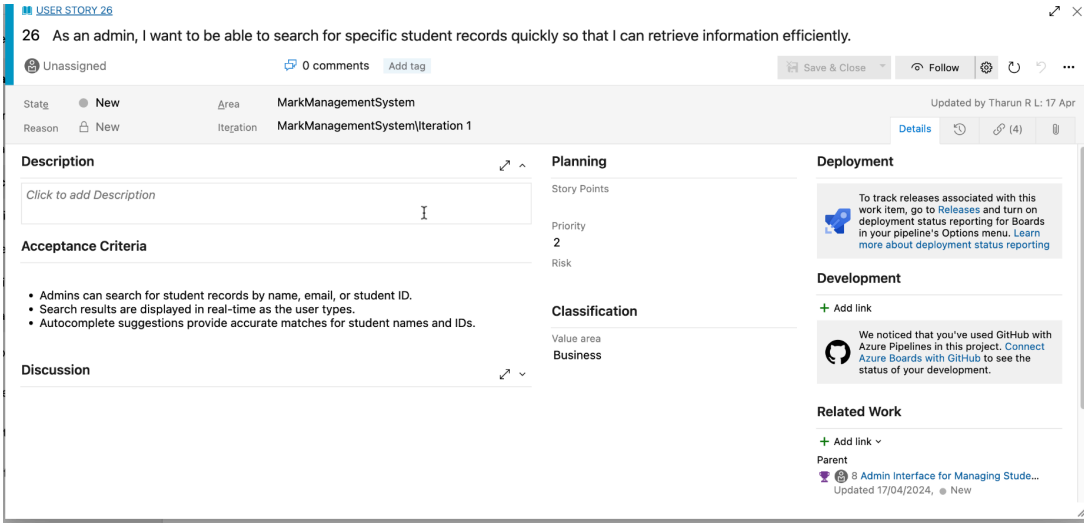
- Comparative statistics, such as class averages and percentile ranks, are displayed alongside student performance data.
- Users can hover over data points to view additional information, such as class average and deviation.
- Class average comparisons are updated dynamically as new data becomes available.

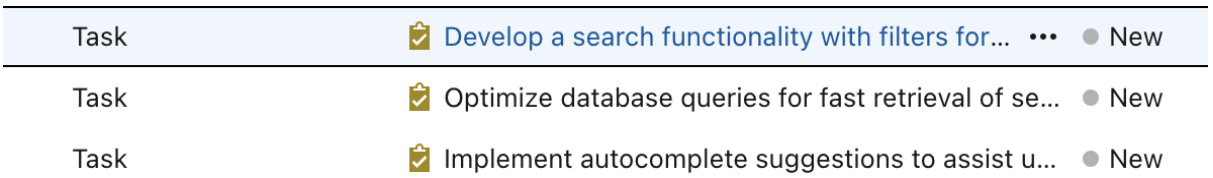

MarkManagementSystem Team
Backlog
Analytics
New Work Item
View as Board
Column Options
Epics
Order
Work Item Type
Title
State
Effort
Busin...
Value Area
Tags
1 Epic Student Data Management New Business
2 Epic User Authentication and Authorization New Business
3 Epic Faculty Mark Management New Business
4 Epic Student Mark Viewing New Business
5 Epic Mark Analysis New Business

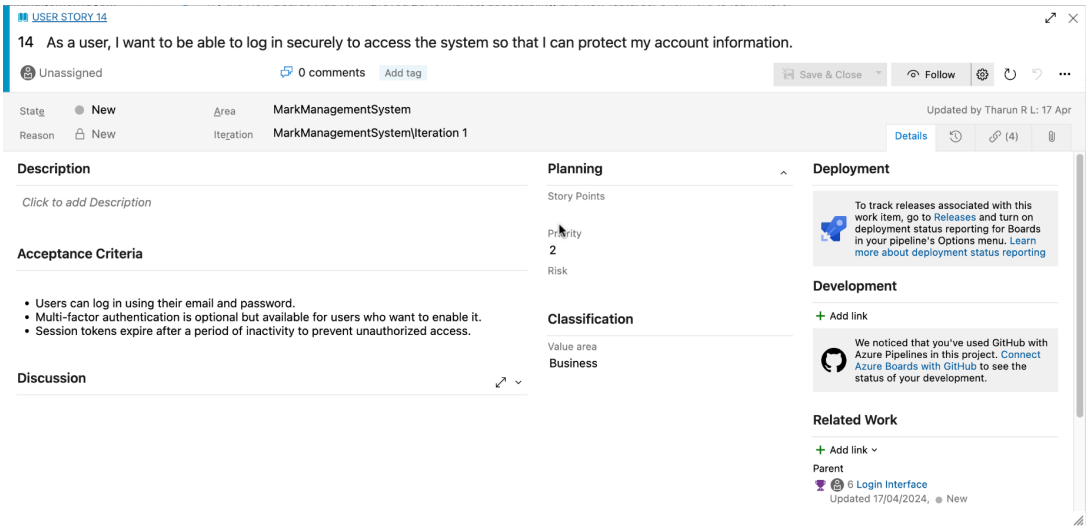

MarkManagementSystem Team
Backlog
Analytics
New Work Item
View as Board
Column Options
Epics
Order
Work Item Type
Title
State
Effort
Busin...
Value Area
Tags
1 Epic Student Data Management New Business
Feature Admin Interface for Managing Student Data New Business
2 Epic User Authentication and Authorization New Business
Feature Login Interface New Business
Feature Access Control New Business
3 Epic Faculty Mark Management New Business
Feature Mark Validation and Submission New Business
Feature Faculty Interface for managing Marks New Business
4 Epic Student Mark Viewing New Business
Feature Student Interface for Viewing Marks New Business
5 Epic Mark Analysis New Business
Feature Analysis Interface for Students New Business
Feature Graphical Representation of Marks New Business

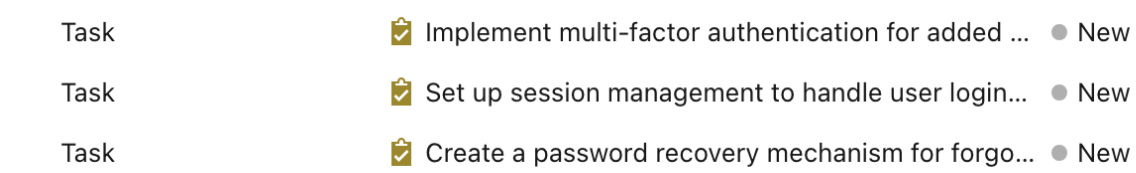

USER STORY 22
22 As an admin, I want to be able to manage student records so that I can keep the database up to date.
Unassigned
0 comments
Add tag
Save & Close
Follow
State New
Reason New
Area MarkManagementSystem
Iteration MarkManagementSystem\Iteration 1
Updated by Tharun R L: 17 Apr
Details
Description
Click to add Description
Acceptance Criteria
Admins can add new student records with required information such as name, email, and student ID.
Data validation prevents the addition of duplicate student records and ensures data consistency.
Discussion
Planning
Story Points
Priority
2
Risk
Classification
Value area
Business
Deployment
To track releases associated with this work item, go to Releases and turn on deployment status reporting for Boards in your pipeline's Options menu. Learn more about deployment status reporting
Development
Add link
We noticed that you've used GitHub with Azure Pipelines in this project. Connect Azure Boards with GitHub to see the status of your development.
Related Work
Add link
Parent
8 Admin Interface for Managing Stude...
Updated 17/04/2024, New

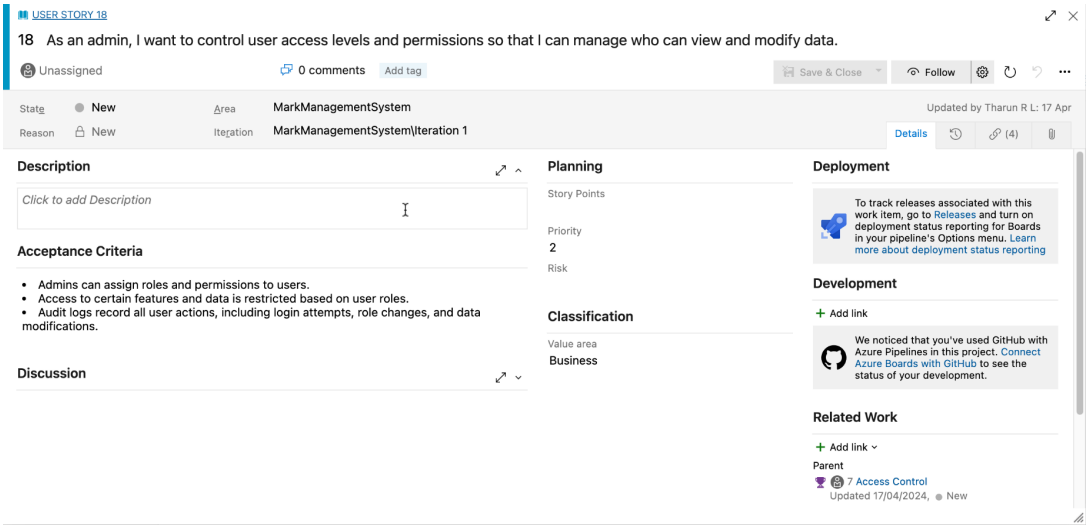

User Story As an admin, I want to be able to manage stude... New
Task Create a user-friendly form for adding new stude... New
Task Implement validation checks to ensure data integ... New
Task Set up database triggers to handle cascading del... New


USER STORY 26
26 As an admin, I want to be able to search for specific student records quickly so that I can retrieve information efficiently.
Unassigned
0 comments
Add tag
Save & Close
Follow
State New
Reason New
Area MarkManagementSystem
Iteration MarkManagementSystem\Iteration 1
Updated by Tharun R L: 17 Apr
Details
(4)
Description
Click to add Description
Acceptance Criteria
• Admins can search for student records by name, email, or student ID.
• Search results are displayed in real-time as the user types.
• Autocomplete suggestions provide accurate matches for student names and IDs.
Discussion
Planning
Story Points
Priority
2
Risk
Classification
Value area
Business
Deployment
To track releases associated with this work item, go to Releases and turn on deployment status reporting for Boards in your pipeline's Options menu. Learn more about deployment status reporting
Development
Add link
We noticed that you've used GitHub with Azure Pipelines in this project. Connect Azure Boards with GitHub to see the status of your development.
Related Work
Add link
Parent
8 Admin Interface for Managing Stude...
Updated 17/04/2024, New

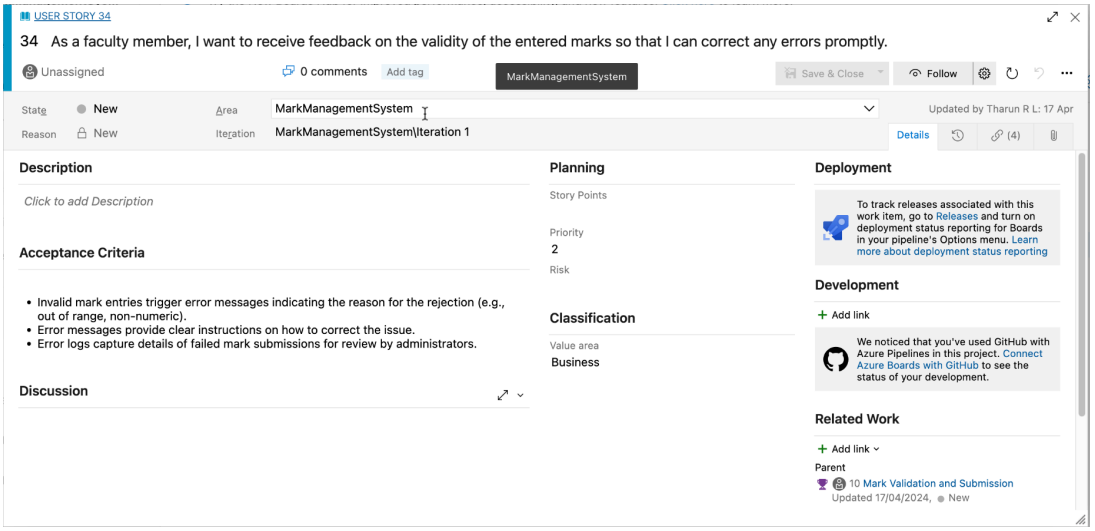

Task Develop a search functionality with filters for... New
Task Optimize database queries for fast retrieval of se... New
Task Implement autocomplete suggestions to assist u... New


USER STORY 14
14 As a user, I want to be able to log in securely to access the system so that I can protect my account information.
Unassigned
0 comments
Add tag
Save & Close
Follow
State New
Reason New
Area MarkManagementSystem
Iteration MarkManagementSystem\Iteration 1
Updated by Tharun R L: 17 Apr
Details
(4)
Description
Click to add Description
Acceptance Criteria
• Users can log in using their email and password.
• Multi-factor authentication is optional but available for users who want to enable it.
• Session tokens expire after a period of inactivity to prevent unauthorized access.
Discussion
Planning
Story Points
Priority
2
Risk
Classification
Value area
Business
Deployment
To track releases associated with this work item, go to Releases and turn on deployment status reporting for Boards in your pipeline's Options menu. Learn more about deployment status reporting
Development
Add link
We noticed that you've used GitHub with Azure Pipelines in this project. Connect Azure Boards with GitHub to see the status of your development.
Related Work
Add link
Parent
6 Login Interface
Updated 17/04/2024, New

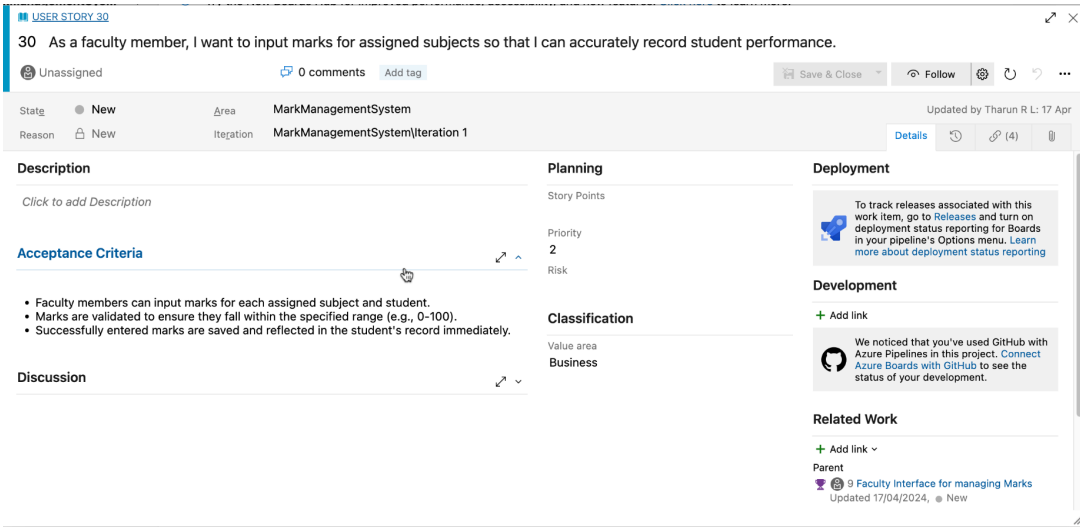

Task Implement multi-factor authentication for added ... New
Task Set up session management to handle user login... New
Task Create a password recovery mechanism for forgo... New



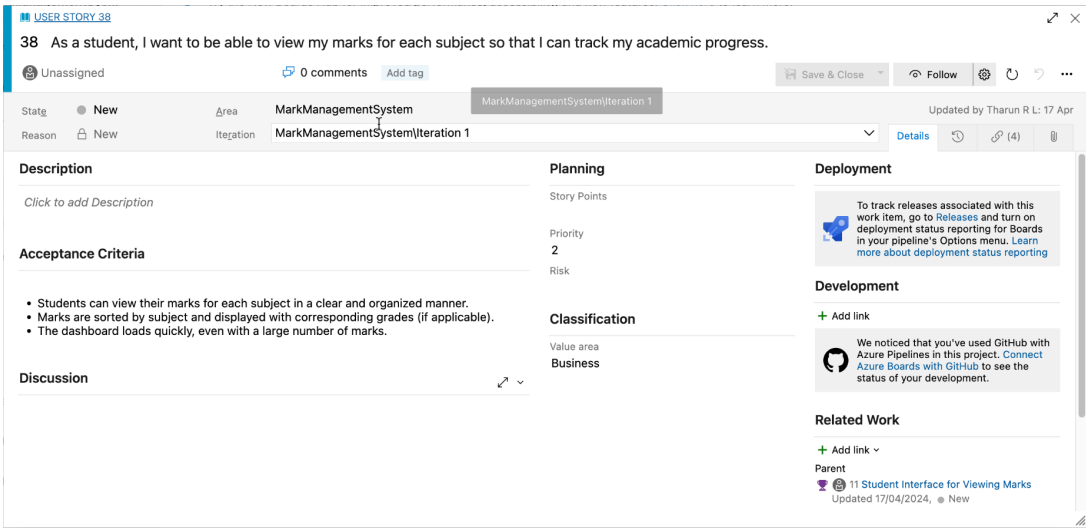
Task	Develop role-based access control (RBAC) syste...	New
Task	Design an admin dashboard to manage user role...	New
Task	Implement auditing functionality to track user act...	New



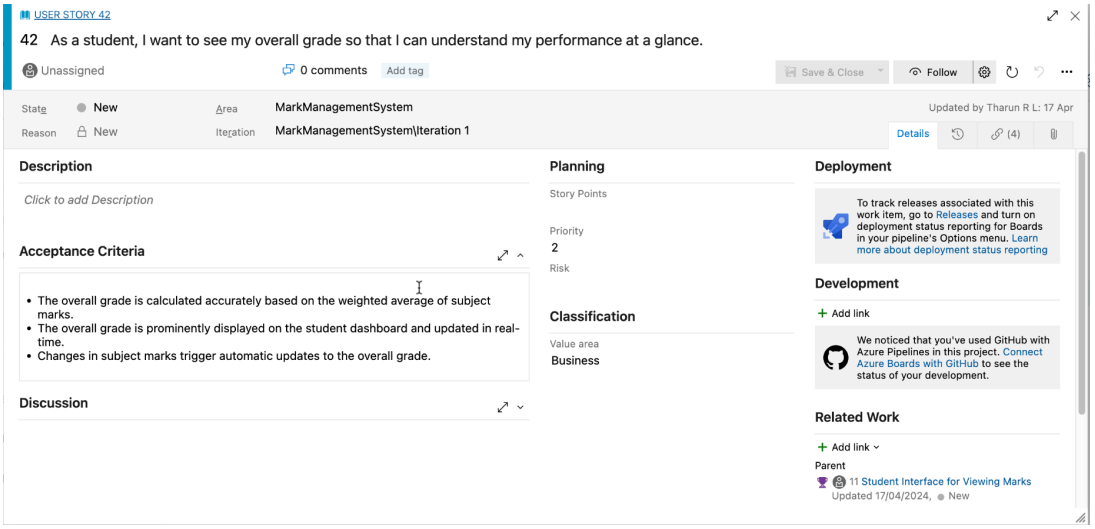
Task	Develop validation rules to check for anomalies i...	New
Task	Implement real-time feedback on invalid entries ...	New
Task	Create an error log for tracking and resolving iss...	New



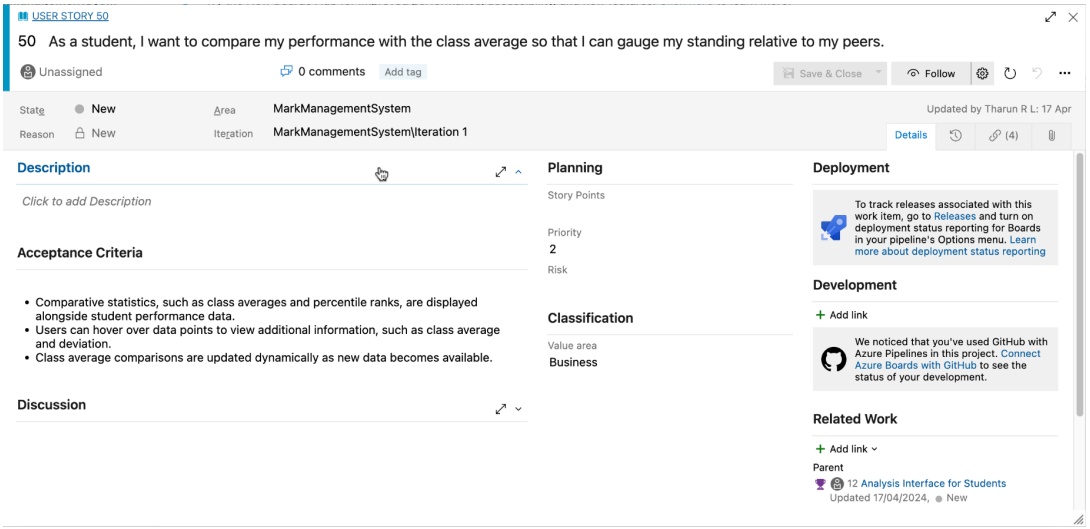
Task	Design a user interface with intuitive input fields ...	New
Task	Implement data validation to ensure entered mar...	New
Task	Create a save mechanism to persist entered mar...	New



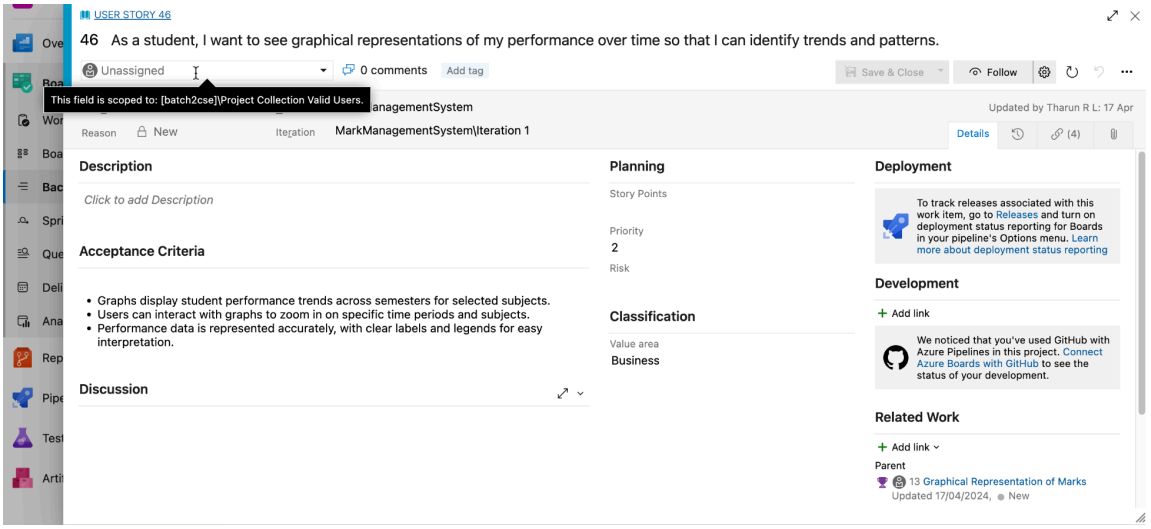
Task	Design a personalized dashboard to display indiv...	New
Task	Implement sorting and filtering options for better...	New
Task	Optimize page loading speed for a seamless user...	New



Task	Develop a grade calculation algorithm based on ...	New
Task	Display the calculated overall grade prominently ...	New
Task	Implement dynamic grade updates upon change...	New



Task	Retrieve and calculate class average mark data f...	New
Task	Display comparative statistics, such as percentil...	New
Task	Implement tooltips or hover effects to provide ad...	New



Task	Integrate a charting library to visualize historical ...	New
Task	Design interactive graphs for displaying perform...	New
Task	Implement zoom and pan functionalities for expl...	New

Estimating Project Effort using Poker Planning Methodology

Poker planning, also known as Planning Poker, is a consensus-based technique for estimating the effort or relative size of user stories in agile development. The process involves team members discussing each story and then independently selecting an estimate card. The team then discusses the estimates to reach a consensus.

Below is a breakdown of the estimation process for the Student Grade Management System (SGMS) project using Poker Planning. We'll identify key user stories and provide estimated story points for each.

Key User Stories and Estimates

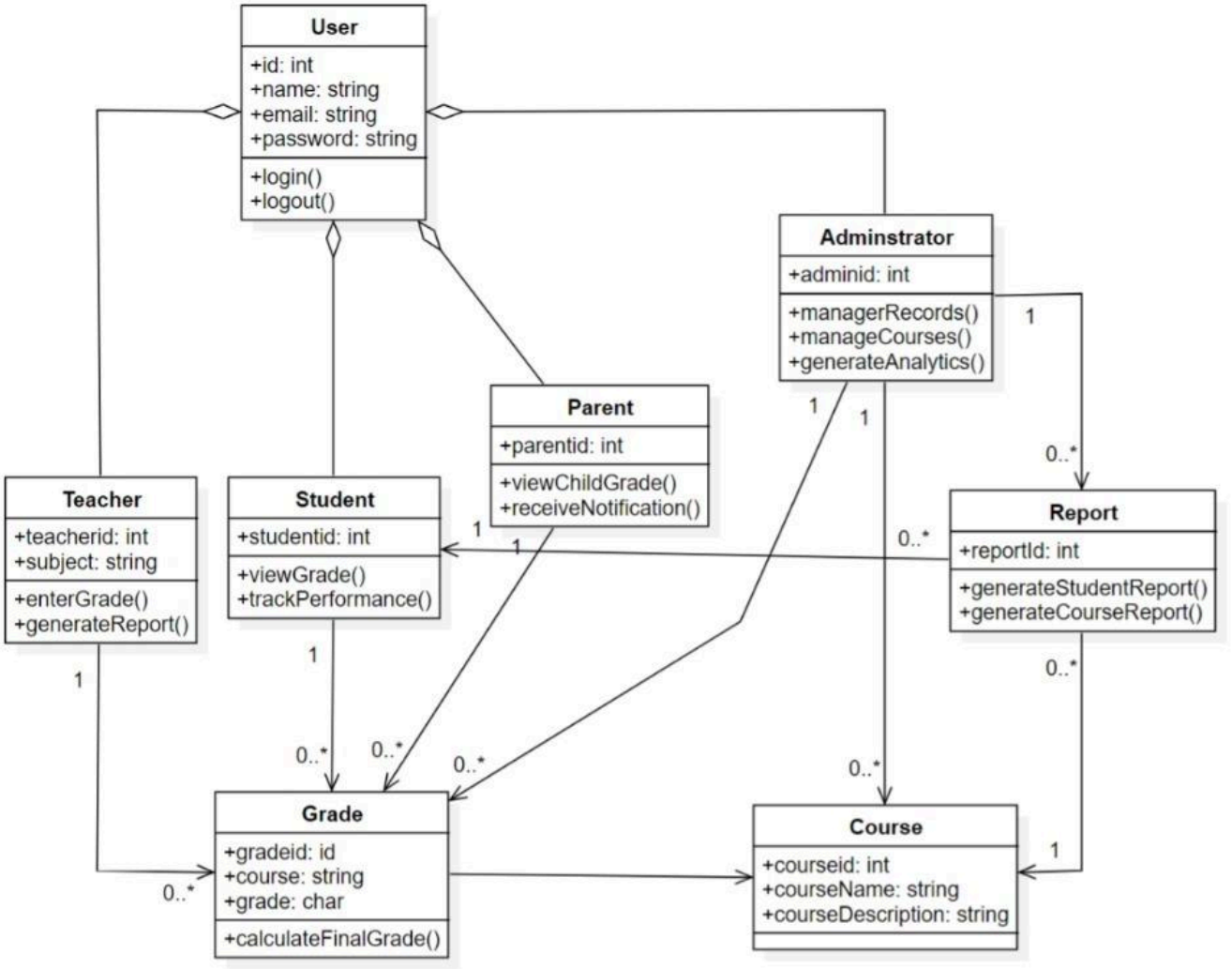
1. **User Authentication and Authorization**
   - **Description**: Implement user login, registration, and role-based access control.
   - **Estimated Story Points**: 8
2. **Teacher Module**
   - **Description**:
     - **Grade Entry**: Interface for entering and updating student grades.
     - **Automated Calculations**: Automate the calculation of final grades based on predefined criteria.
     - **Report Generation**: Tools for generating and exporting grade reports.
   - **Estimated Story Points**: 20
3. **Administrator Module**
   - **Description**:
     - **Student Information Management**: Manage student records, enrollments, and course registrations.
     - **Data Integrity Checks**: Implement validation and verification mechanisms for data accuracy.

        - **Analytics and Reporting**: Advanced tools for generating detailed performance reports and analysis.
    - **Estimated Story Points**: 15
4. **Student Module**
    - **Description**:
        - **Real-Time Access**: Interface for students to view their grades and academic progress.
        - **Performance Tracking**: Tools for tracking academic performance over time.
    - **Estimated Story Points**: 13
5. **Parent Module**
    - **Description**:
        - **Real-Time Notifications**: System for notifying parents about their child's grades and academic issues.
        - **Access to Reports**: Interface for parents to view detailed performance reports.
    - **Estimated Story Points**: 8
6. **Database Design and Management**
    - **Description**:
        - **Database Schema**: Design the database schema to support all modules.
        - **Centralized Storage**: Implement a centralized storage solution for all academic records.
    - **Estimated Story Points**: 10
7. **Security and Privacy**
    - **Description**: Implement security measures to protect sensitive data and ensure compliance with privacy regulations.
    - **Estimated Story Points**: 8
8. **User Interface Design**
    - **Description**:
        - **Design Interfaces**: Develop user-friendly interfaces for teachers, administrators, students, and parents.
        - **User Experience**: Ensure a consistent and intuitive user experience across all modules.
    - **Estimated Story Points**: 13
9. **Integration with Existing Systems**
    - **Description**: Integrate the SGMS with existing school management systems and other relevant software.
    - **Estimated Story Points**: 10
10. **Testing and Quality Assurance**
    - **Description**: Comprehensive testing of all modules to ensure functionality, performance, and security.
    - **Estimated Story Points**: 15
11. **Deployment and Training**
    - **Description**: Deploy the system to production and provide training to users.
    - **Estimated Story Points**: 8

Total Estimated Story Points: 128

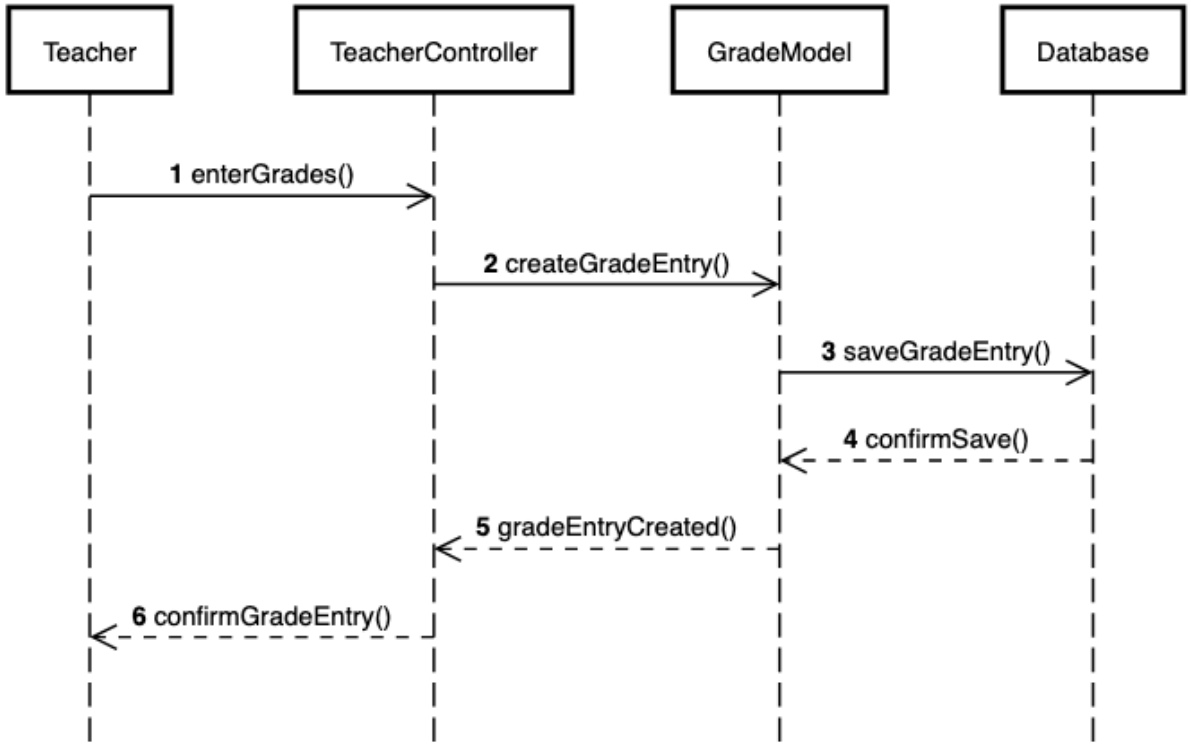
Architecture Diagram

Class diagram

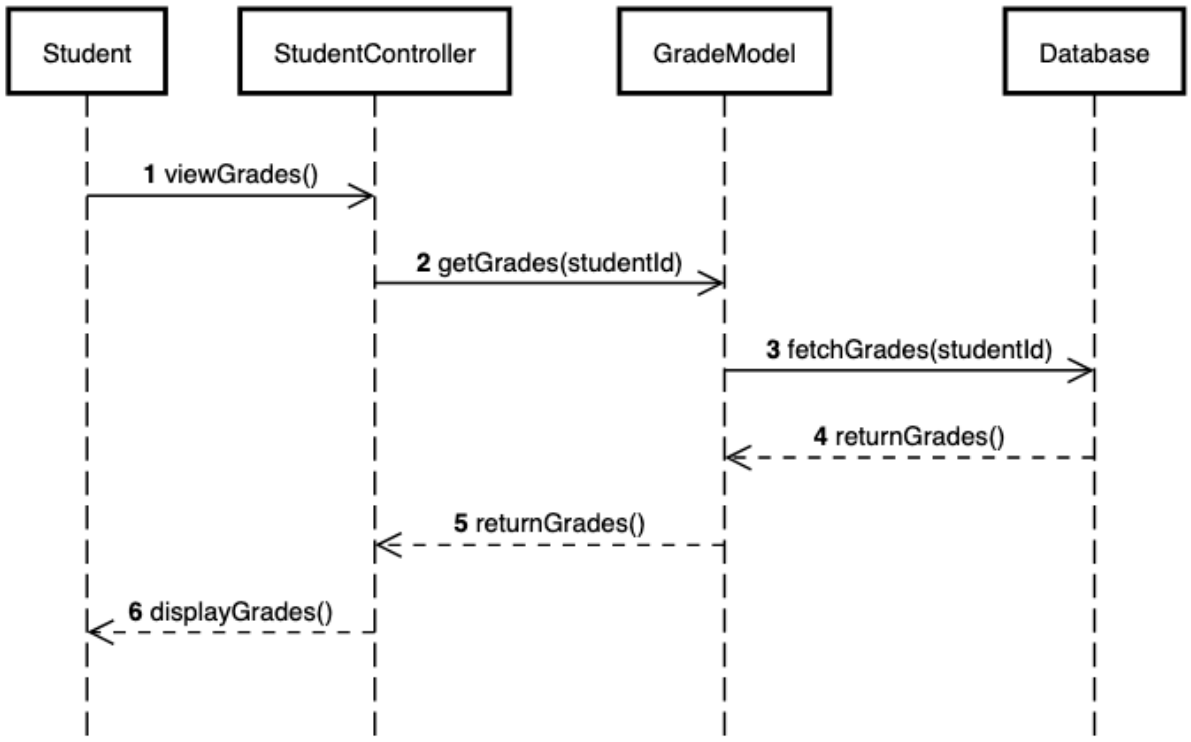


Sequence diagram

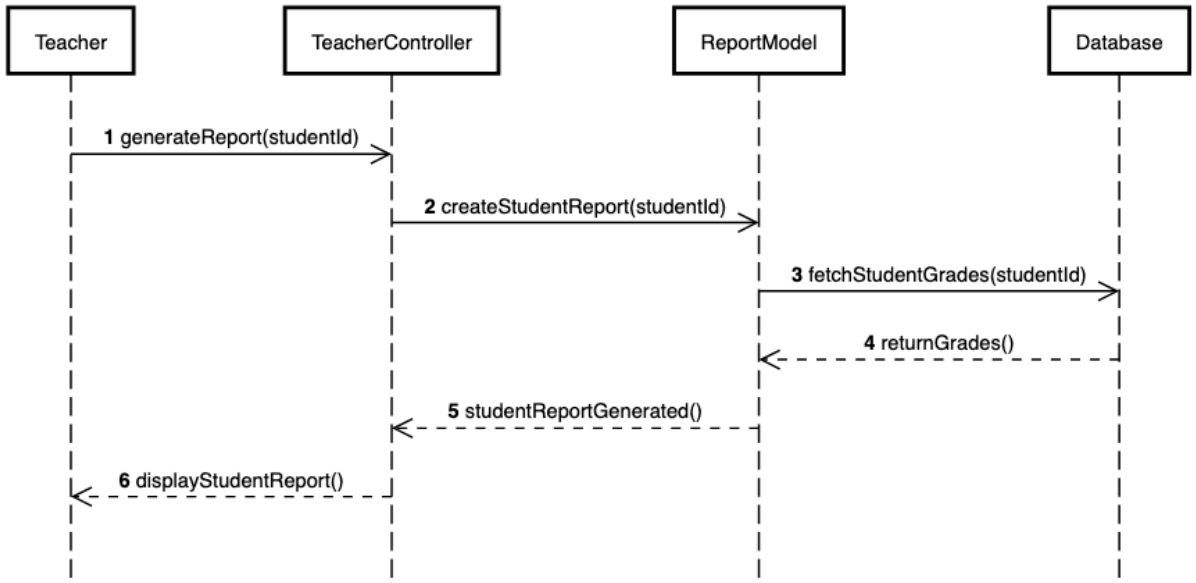
User Story 1: Teacher Enters Grades



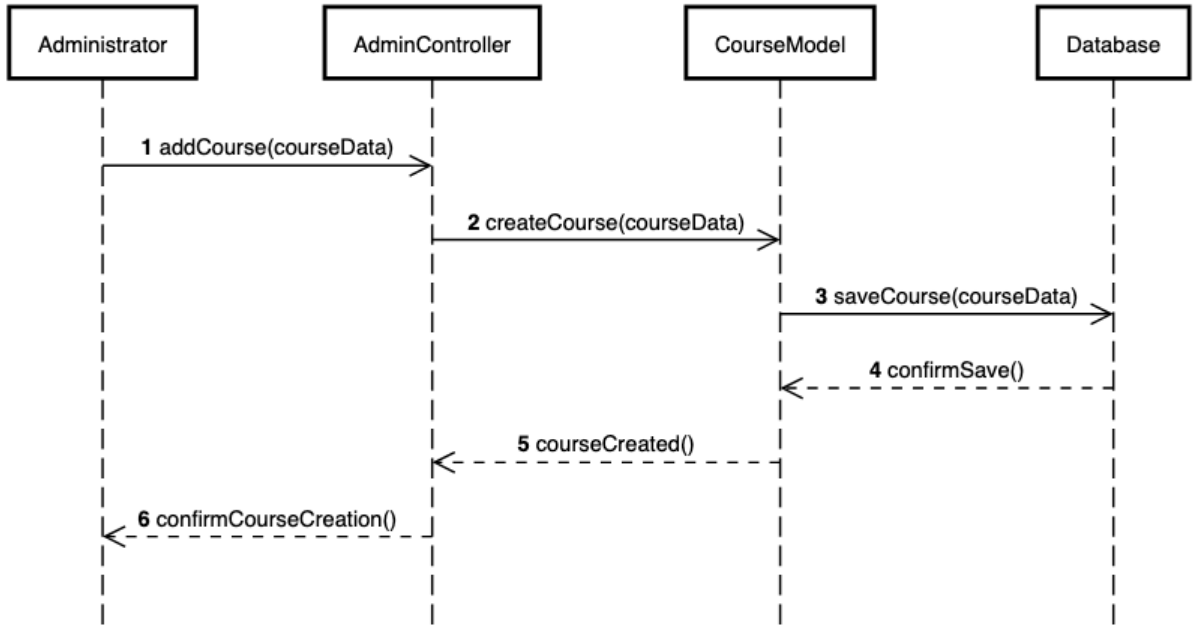
User Story 2: Student Views Grades



User Story 3: Teacher Generates Report



User Story 4: Administrator Manages Courses



Test Strategy

Objective
The objective of the testing strategy for the Student Grade Management System (SGMS) is to ensure the system meets all functional and non-functional requirements, is free of defects, and provides a high-quality user experience.
Scope
The testing strategy covers the following aspects:

- Functional Testing
- Integration Testing
- Performance Testing
- Security Testing
- User Acceptance Testing (UAT)

Tools

- **Unit Testing**: JUnit
- **Integration Testing**: Postman
- **Performance Testing**: Apache JMeter
- **Security Testing**: OWASP ZAP
- **Continuous Integration**: Azure DevOps Pipelines
- **Version Control**: GitHub

Test Plans
Functional Testing Plan

1. **Test Objective**: Verify that all user stories are implemented as expected.
2. **Test Scope**: All functionalities covered by user stories.
3. **Test Approach**: Use manual and automated testing techniques to validate functionality.
4. **Tools**: JUnit, Postman
5. **Test Cases**: Detailed below.

Integration Testing Plan

1. **Test Objective**: Ensure all components of the SGMS integrate seamlessly.
2. **Test Scope**: Integration points between Controllers, Models, Views, and Database.
3. **Test Approach**: Validate data flow and functionality across integrated components.
4. **Tools**: Postman, JUnit

Performance Testing Plan

1. **Test Objective**: Validate system performance under various conditions.
2. **Test Scope**: Response times, throughput, and resource utilization.
3. **Test Approach**: Simulate load and measure performance metrics.
4. **Tools**: Apache JMeter

Security Testing Plan

1. **Test Objective**: Identify and fix security vulnerabilities.
2. **Test Scope**: All modules of the SGMS.

3. **Test Approach**: Conduct penetration testing and vulnerability scanning.
4. **Tools**: OWASP ZAP

User Acceptance Testing Plan

1. **Test Objective**: Validate the system with real users.
2. **Test Scope**: All user stories.
3. **Test Approach**: Conduct testing sessions with end-users.
4. **Tools**: Manual Testing

Test Cases
User Story 1: Teacher Enters Grades
Happy Path:

1. **Test Case**: Enter valid grades
    - **Precondition**: Teacher is logged in.
    - **Steps**:
        1. Navigate to the grade entry page.
        2. Enter valid student ID and grade.
        3. Submit the form.
    - **Expected Result**: Grade is successfully entered, and a confirmation message is displayed.

Error Scenarios:

1. **Test Case**: Enter invalid grades
    - **Precondition**: Teacher is logged in.
    - **Steps**:
        1. Navigate to the grade entry page.
        2. Enter an invalid grade (e.g., a letter instead of a number).
        3. Submit the form.
    - **Expected Result**: An error message is displayed indicating invalid grade format.
2. **Test Case**: Missing student ID
    - **Precondition**: Teacher is logged in.
    - **Steps**:
        1. Navigate to the grade entry page.
        2. Leave the student ID field empty.
        3. Submit the form.
    - **Expected Result**: An error message is displayed indicating the student ID is required.

User Story 2: Student Views Grades
Happy Path:

1. **Test Case**: View grades
    - **Precondition**: Student is logged in.
    - **Steps**:
        1. Navigate to the grades page.
        2. View the list of grades.
    - **Expected Result**: Grades are displayed correctly.

Error Scenarios:

1. **Test Case**: No grades available
    - **Precondition**: Student is logged in.
    - **Steps**:
        1. Navigate to the grades page.
    - **Expected Result**: A message is displayed indicating no grades are available.

User Story 3: Teacher Generates Reports
Happy Path:

1. **Test Case**: Generate student report
    - **Precondition**: Teacher is logged in.
    - **Steps**:
        1. Navigate to the report generation page.
        2. Select a student.
        3. Generate the report.
    - **Expected Result**: Report is successfully generated and displayed.

Error Scenarios:

1. **Test Case**: Generate report for invalid student ID
    - **Precondition**: Teacher is logged in.
    - **Steps**:
        1. Navigate to the report generation page.
        2. Enter an invalid student ID.
        3. Submit the form.
    - **Expected Result**: An error message is displayed indicating the student ID is invalid.

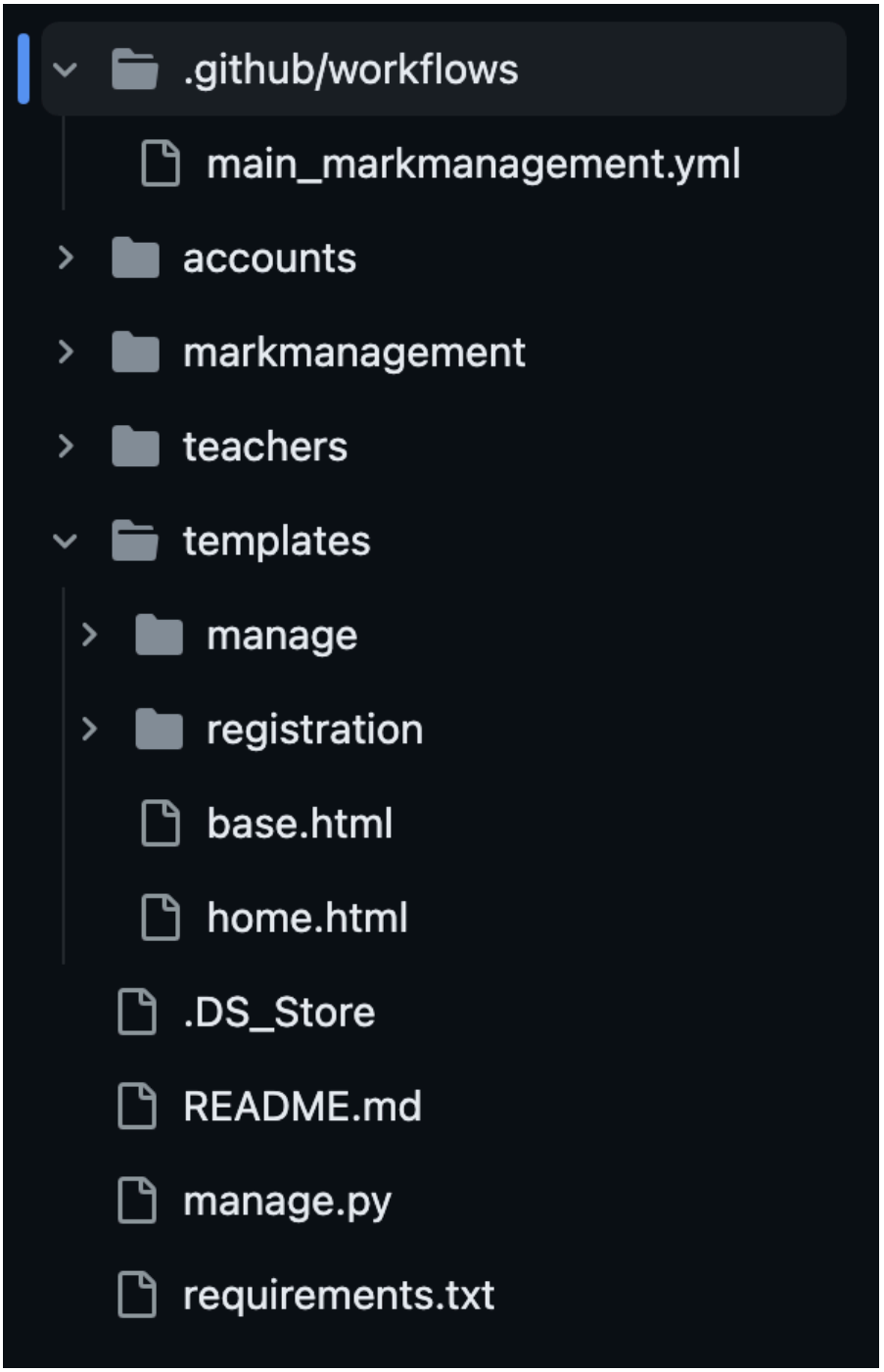
User Story 4: Administrator Manages Courses
Happy Path:

1. **Test Case**: Add a new course
    - **Precondition**: Administrator is logged in.
    - **Steps**:
        1. Navigate to the course management page.
        2. Enter valid course details.
        3. Submit the form.
    - **Expected Result**: Course is successfully added, and a confirmation message is displayed.

Error Scenarios:

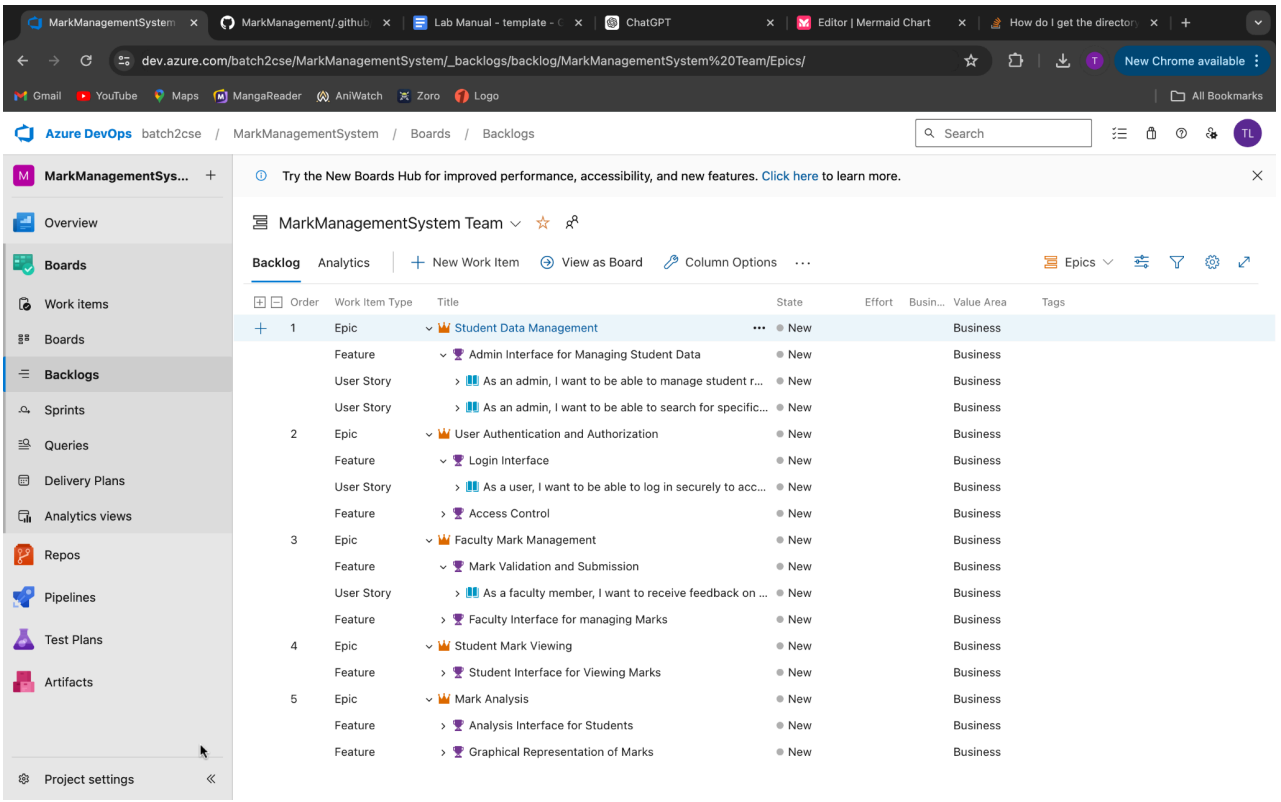
1. **Test Case**: Enter invalid course details
    - **Precondition**: Administrator is logged in.
    - **Steps**:
        1. Navigate to the course management page.
        2. Enter invalid course details (e.g., missing course name).
        3. Submit the form.
    - **Expected Result**: An error message is displayed indicating the required fields.

A view of the github repository showcasing the Project structure, their naming conventions

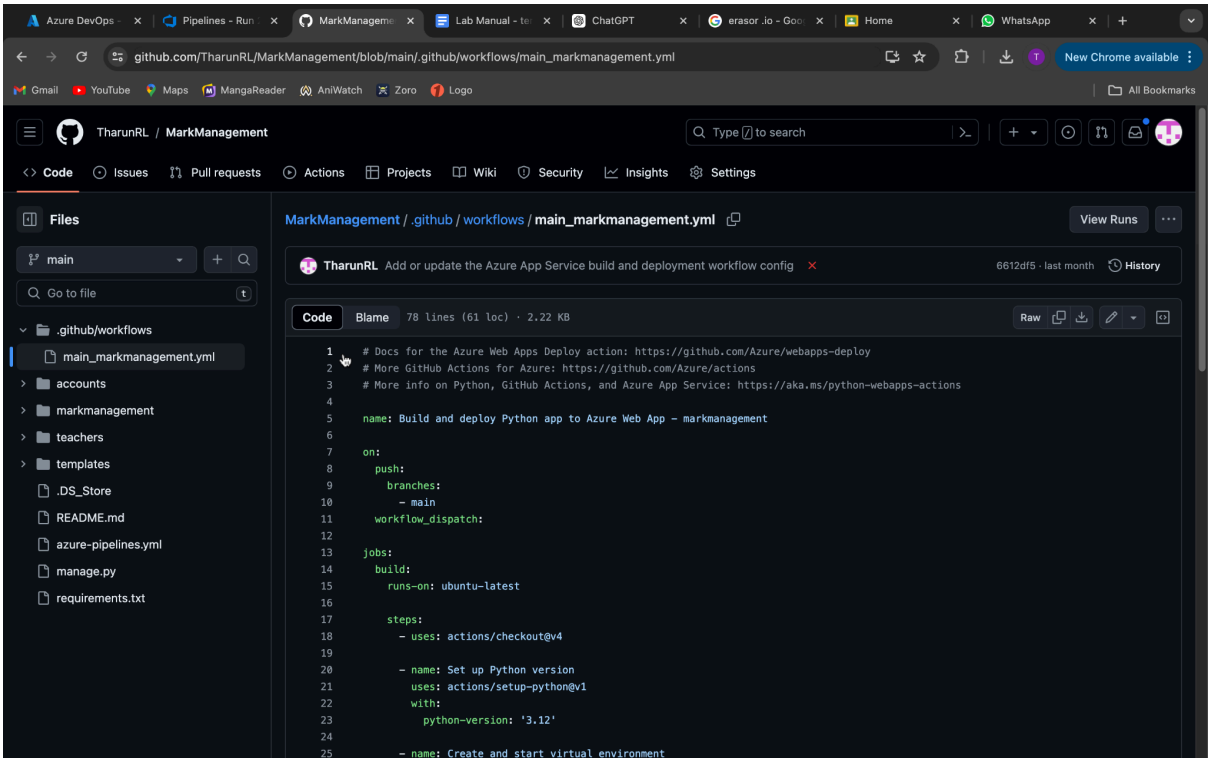


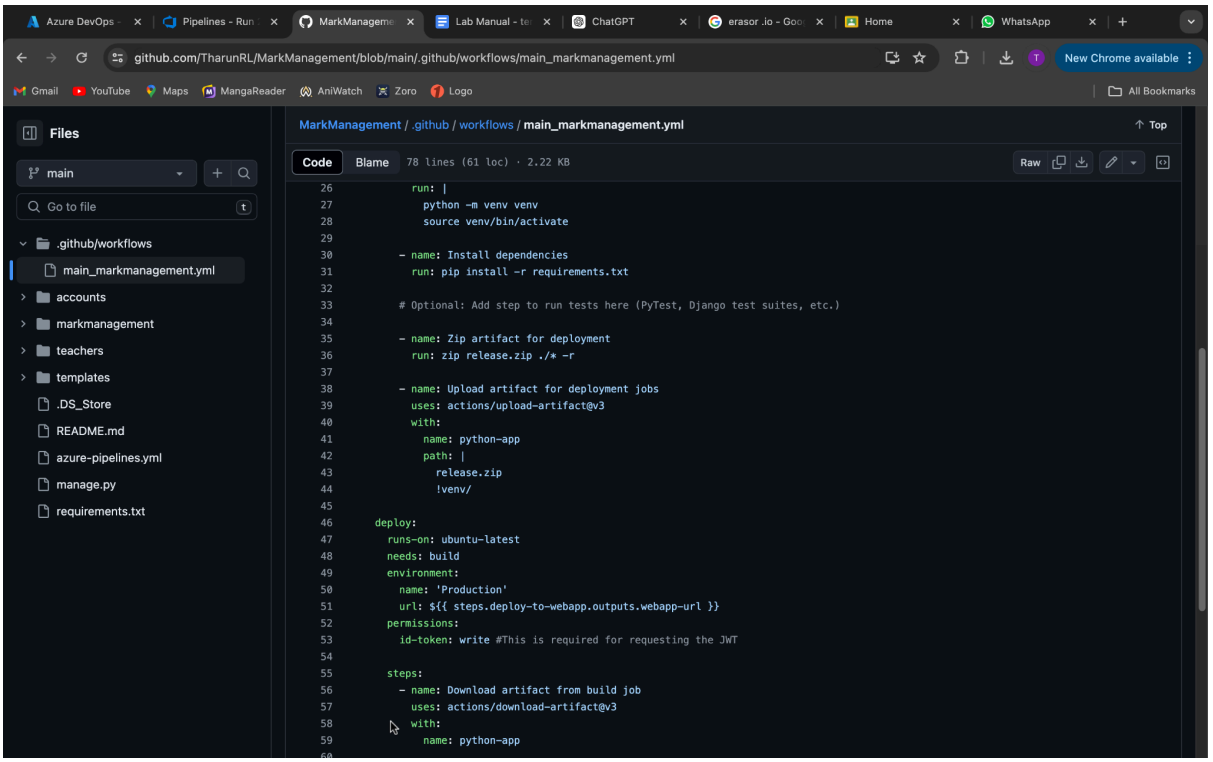
Software Concepts & Engineering - Lab Manual

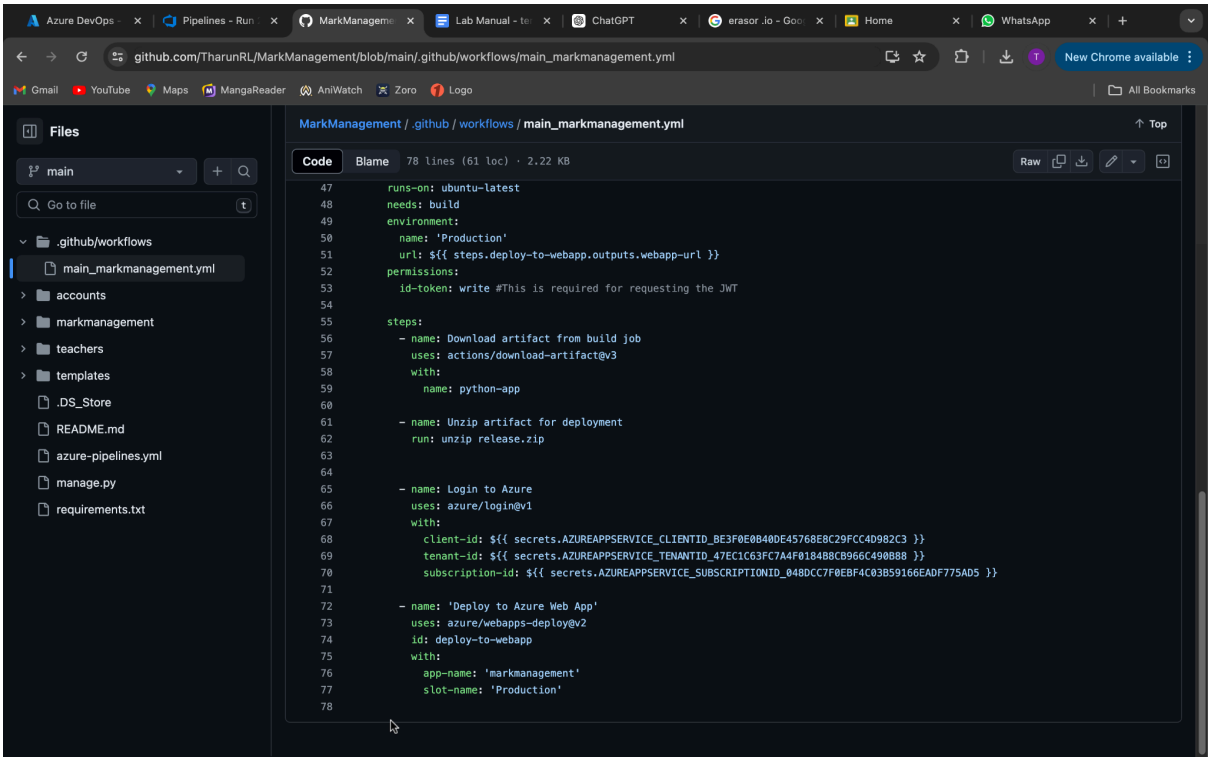
DevOps Architecture in Azure

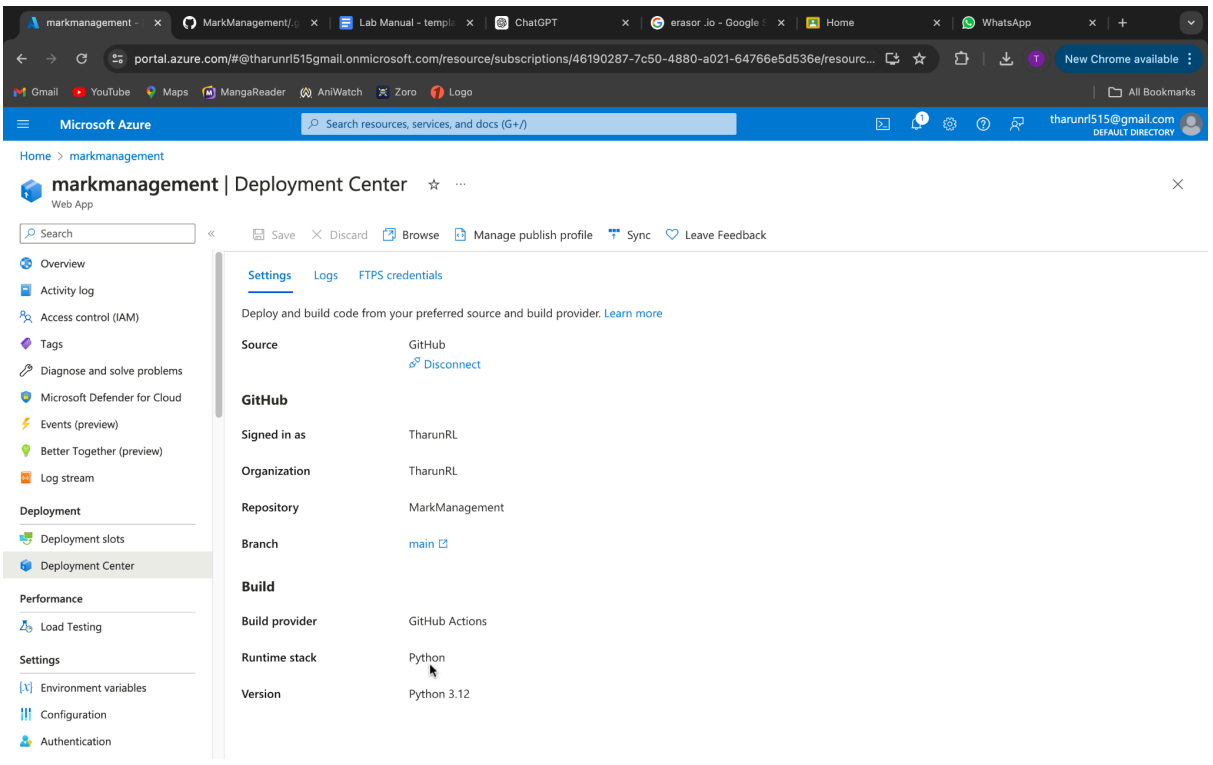


Deployment Architecture of the application

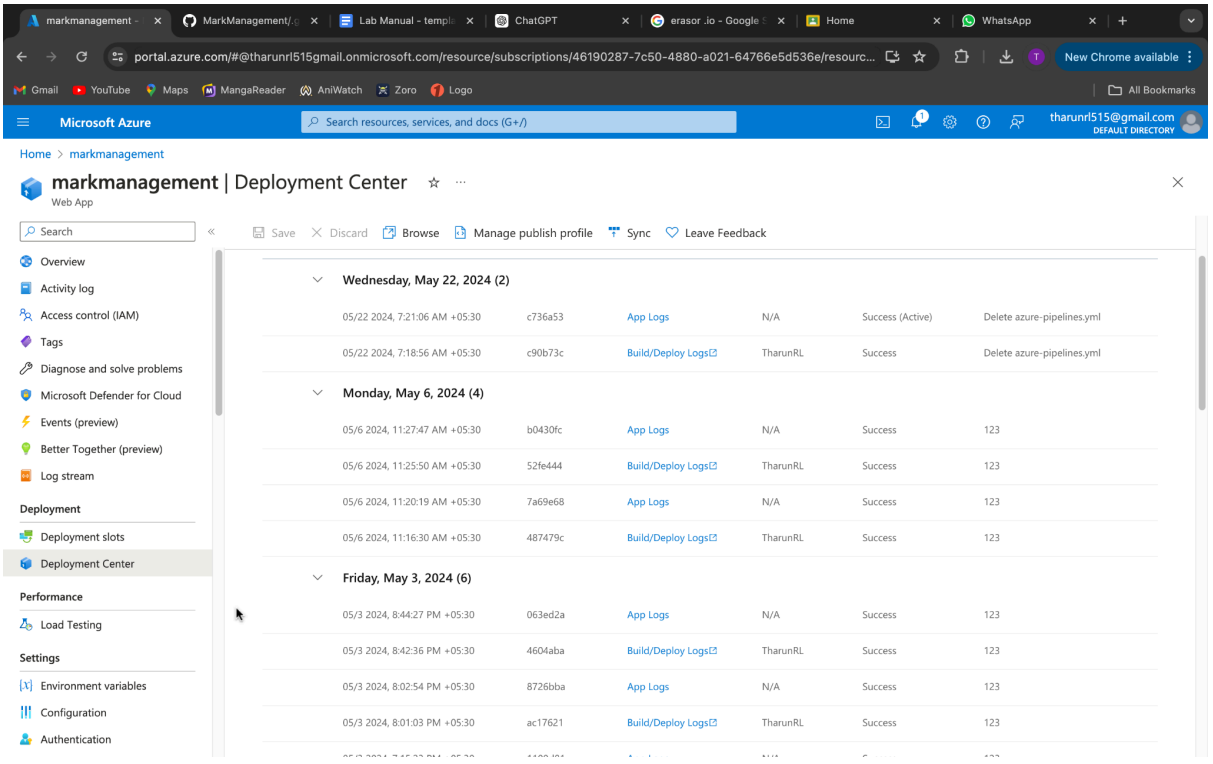


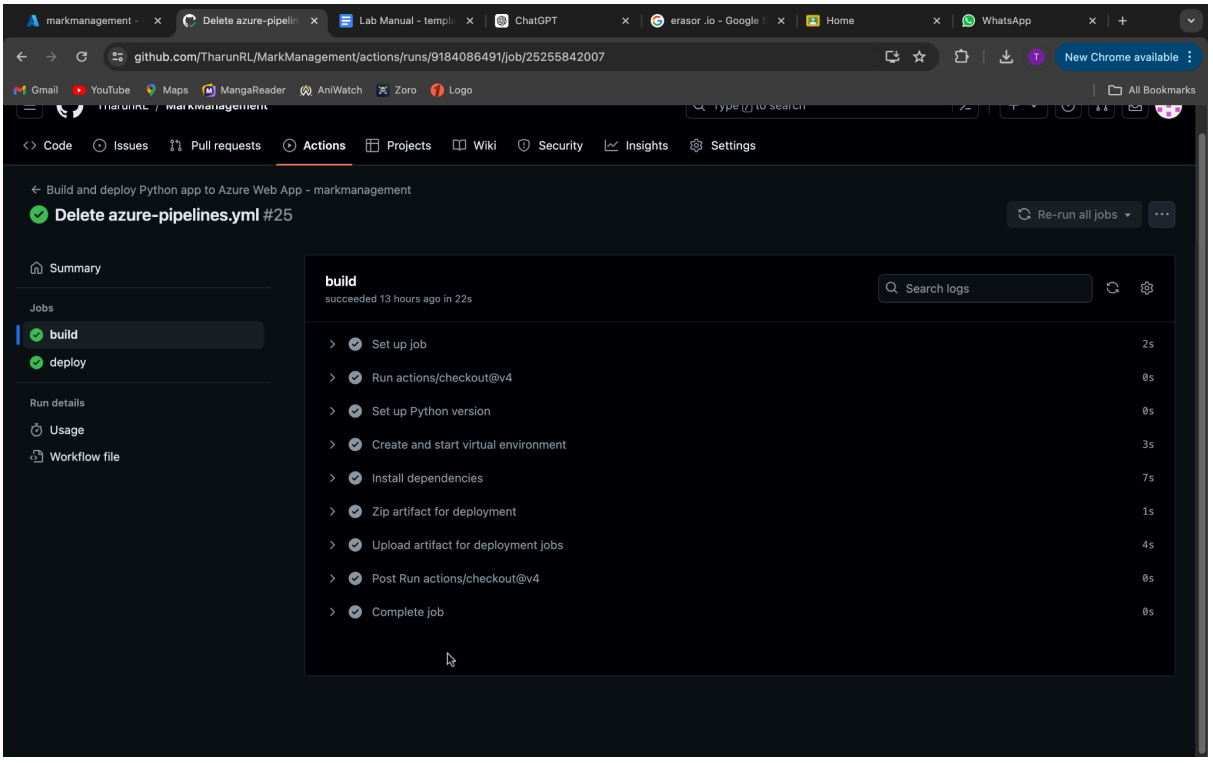


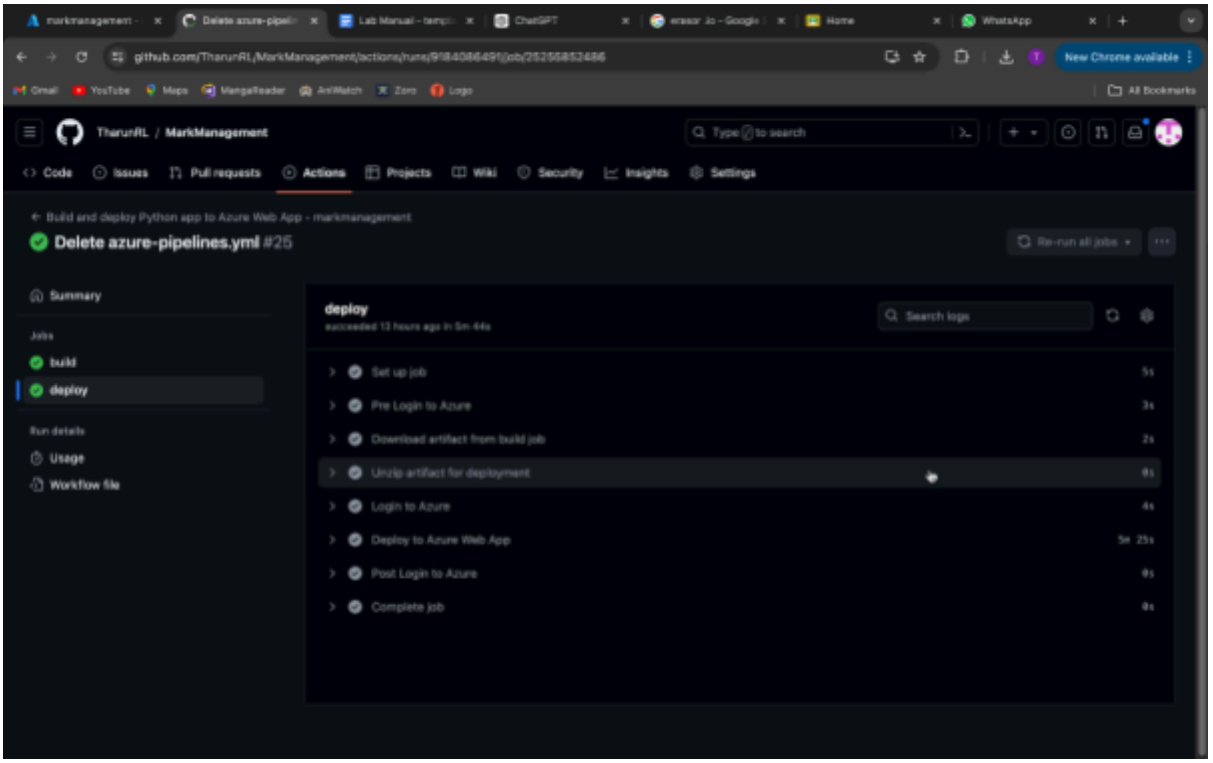

github.com/TharunRL/MarkManagement/blob/main/.github/workflows/main_markmanagement.yml
Files
main
Go to file
.github/workflows
main_markmanagement.yml
accounts
markmanagement
teachers
templates
.DS_Store
README.md
azure-pipelines.yml
manage.py
requirements.txt
MarkManagement / .github / workflows / main_markmanagement.yml
Code
Blame
78 lines (61 loc) · 2.22 KB
Raw
runs-on: ubuntu-latest
needs: build
environment:
name: 'Production'
url: ${{ steps.deploy-to-webapp.outputs.webapp-url }}
permissions:
id-token: write #This is required for requesting the JWT
steps:
- name: Download artifact from build job
uses: actions/download-artifact@v3
with:
name: python-app
- name: Unzip artifact for deployment
run: unzip release.zip
- name: Login to Azure
uses: azure/login@v1
with:
client-id: ${{ secrets.AZUREAPPSERVICE_CLIENTID_BE3F0E0B40DE45768E8C29FCC4D982C3 }}
tenant-id: ${{ secrets.AZUREAPPSERVICE_TENANTID_47EC1C63FC7A4F0184B8CB966C490B88 }}
subscription-id: ${{ secrets.AZUREAPPSERVICE_SUBSCRIPTIONID_048DCC7F0EBF4C03B59166EADF775AD5 }}
- name: 'Deploy to Azure Web App'
uses: azure/webapps-deploy@v2
id: deploy-to-webapp
with:
app-name: 'markmanagement'
slot-name: 'Production'

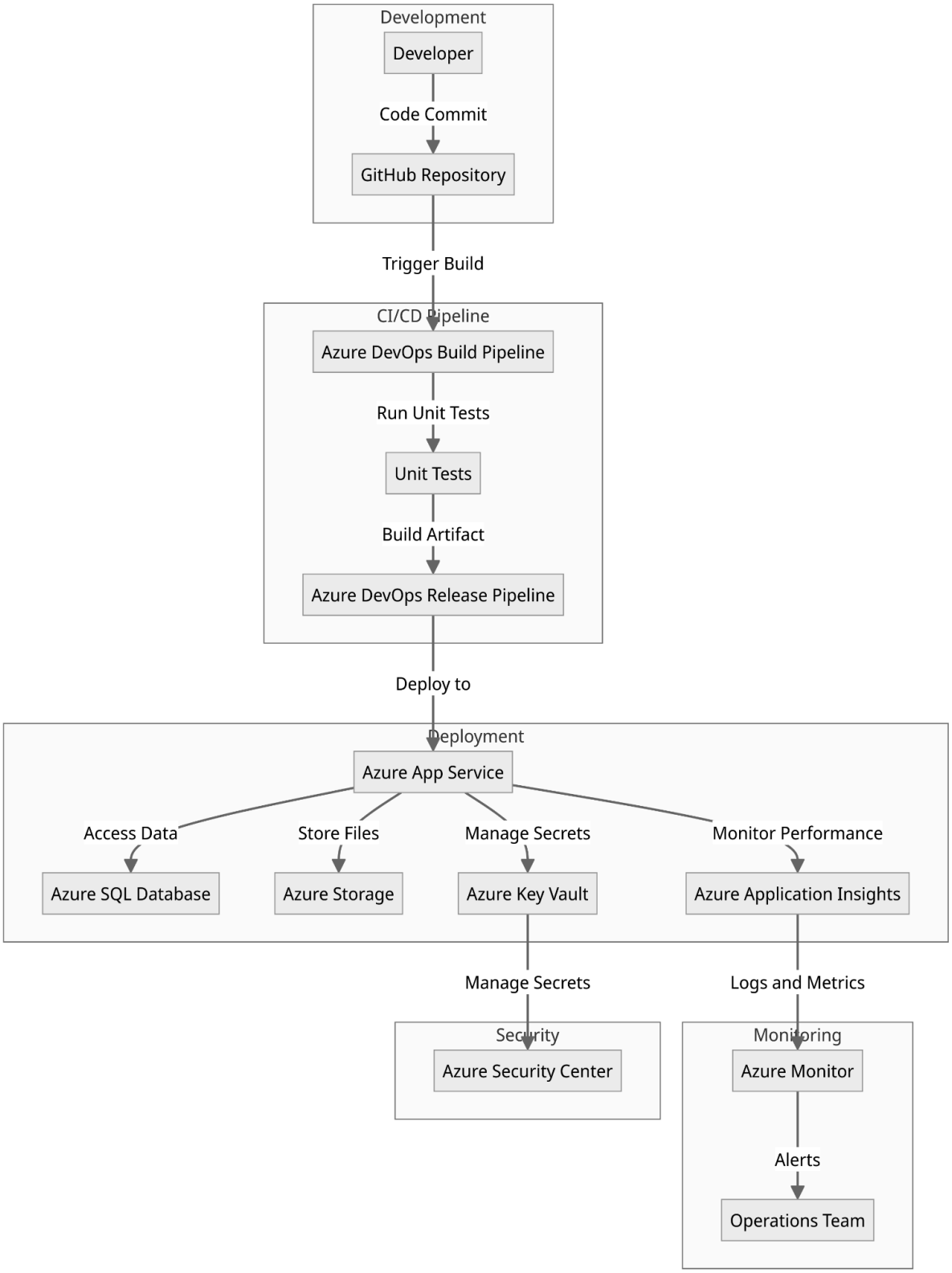

portal.azure.com/#@tharunrl515gmail.onmicrosoft.com/resource/subscriptions/46190287-7c50-4880-a021-64766e5d536e/resourc...
Microsoft Azure
Search resources, services, and docs (G+/)
tharunrl515@gmail.com
DEFAULT DIRECTORY
Home > markmanagement
markmanagement | Deployment Center
Web App
Search
Save
Discard
Browse
Manage publish profile
Sync
Leave Feedback
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Microsoft Defender for Cloud
Events (preview)
Better Together (preview)
Log stream
Deployment
Deployment slots
Deployment Center
Performance
Load Testing
Settings
Environment variables
Configuration
Authentication
Settings
Logs
FTPS credentials
Deploy and build code from your preferred source and build provider. Learn more
Source
GitHub
Disconnect
GitHub
Signed in as
TharunRL
Organization
TharunRL
Repository
MarkManagement
Branch
main
Build
Build provider
GitHub Actions
Runtime stack
Python
Version
Python 3.12

Software Concepts & Engineering - Lab Manual






github.com/TharunRL/MarkManagement/actions/runs/9184086491/job/25256852486
TharunRL / MarkManagement
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Build and deploy Python app to Azure Web App - markmanagement
Delete azure-pipelines.yml #25
Summary
Jobs
build
deploy
Run details
Usage
Workflow file
deploy
Set up job
Pre Login to Azure
Download artifact from build job
Unzip artifact for deployment
Login to Azure
Deploy to Azure Web App
Post Login to Azure
Complete job


Development
Developer
Code Commit
GitHub Repository
Trigger Build
CI/CD Pipeline
Azure DevOps Build Pipeline
Run Unit Tests
Unit Tests
Build Artifact
Azure DevOps Release Pipeline
Deploy to
Deployment
Azure App Service
Access Data
Store Files
Manage Secrets
Monitor Performance
Azure SQL Database
Azure Storage
Azure Key Vault
Azure Application Insights
Manage Secrets
Logs and Metrics
Security
Azure Security Center
Monitoring
Azure Monitor
Alerts
Operations Team

Explanation of the Deployment Architecture

1. **Development**:
    - **Developer**: Writes code and commits it to the **GitHub Repository**.
2. **CI/CD Pipeline**:
    - **GitHub Repository**: Triggers the build pipeline in **Azure DevOps**.
    - **Azure DevOps Build Pipeline**: Compiles the code and runs **Unit Tests**.
    - **Unit Tests**: Validates the code functionality.
    - Successful builds produce a **Build Artifact**.
    - **Azure DevOps Release Pipeline**: Deploys the build artifact to the deployment environment.
3. **Deployment**:
    - **Azure App Service**: Hosts the application and serves it to users.
    - **Azure SQL Database**: Stores application data.
    - **Azure Storage**: Stores files and other data.
    - **Azure Application Insights**: Monitors application performance and collects telemetry data.
    - **Azure Key Vault**: Manages secrets, such as database connection strings and API keys.
4. **Monitoring**:
    - **Azure Application Insights**: Sends logs and performance metrics to **Azure Monitor**.
    - **Azure Monitor**: Provides comprehensive monitoring and alerting capabilities.
    - **Operations Team**: Receives alerts and takes necessary actions.
5. **Security**:
    - **Azure Key Vault**: Ensures secure management of secrets.
    - **Azure Security Center**: Manages security posture and provides recommendations.