**Azure DevOps – Theoretical Assignment**

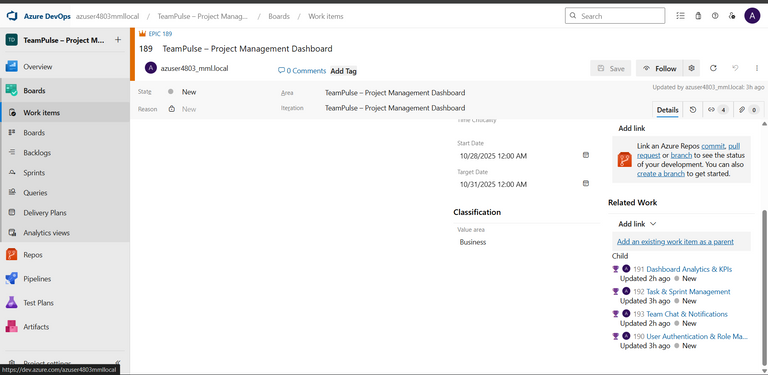
**Problem Statement:**

You are part of a software development team working on a new internal tool named “TeamPulse” — a web-based project management dashboard that helps organizations track productivity, project milestones, and collaboration efficiency.

Your task is to structure this project in Azure DevOps, defining a clear hierarchy of Epics → Features → User Stories → Tasks. This is a theoretical and planning-focused exercise — no actual code is required. All items will be created and visualized in the Azure DevOps Boards

**Epic: TeamPulse – Project Management Dashboard**

**Description :** A web-based project management dashboard that helps organizations track productivity, project milestones, and collaboration efficiency.



**Features (4 total):**

**1. User Authentication & Role Management**

**2. Dashboard Analytics & KPIs**

**3. Task & Sprint Management**

**4. Team Chat & Notifications**

**Feature 1: User Authentication & Role Management**

**User Story 1.1:** As an Admin, I want to create and manage user accounts so that I can control system access.   
 **Tasks:**

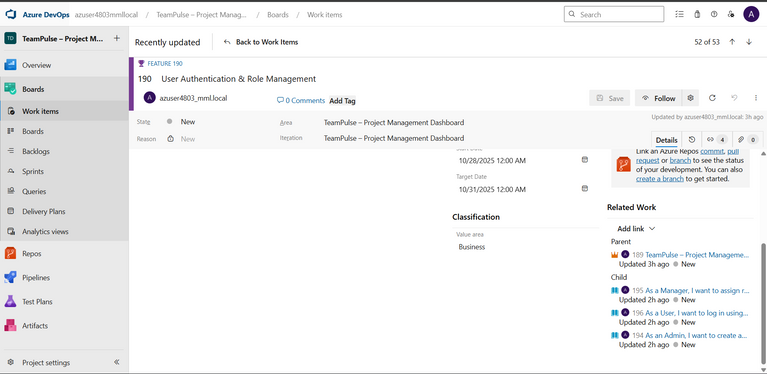
* Define user roles and permissions
* Design user creation forms
* Document account lifecycle

**User Story 1.2:** As a Manager, I want to assign roles to team members so that access rights are well maintained.   
 **Tasks:**

* Map roles to permissions
* Create role assignment workflow
* Test role-based access control scenarios

**User Story 1.3:** As a User, I want to log in using my credentials so that I can access my assigned tasks.   
 **Tasks:**

* Design login and registration forms
* Create password reset logic
* Define error handling for failed logins



**Feature 2: Dashboard Analytics & KPIs**

**User Story 2.1:** As a Manager, I want to view team performance metrics so that I can identify productivity trends.   
 **Tasks:**

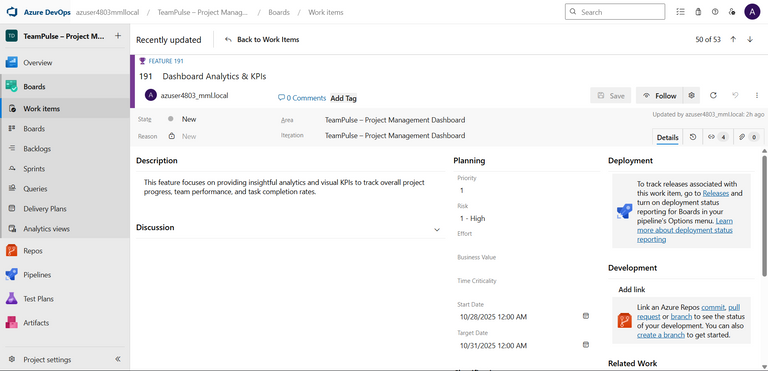
* Identify key performance indicators (KPIs)
* Design dashboard layout
* Document data flow for analytics

**User Story 2.2:** As a User, I want to see my individual task progress so that I can monitor my workload.   
 **Tasks:**

* Create progress calculation logic
* Design progress bar component
* Define data refresh interval

**User Story 2.3:** As an Admin, I want to generate project summary reports so that I can share them with stakeholders.   
 **Tasks:**

* List report types and formats
* Document export workflow
* Create report validation checklist



**Feature 3: Task & Sprint Management**

**User Story 3.1:** As a Manager, I want to create sprints and assign tasks so that I can plan development cycles.   
 **Tasks:**

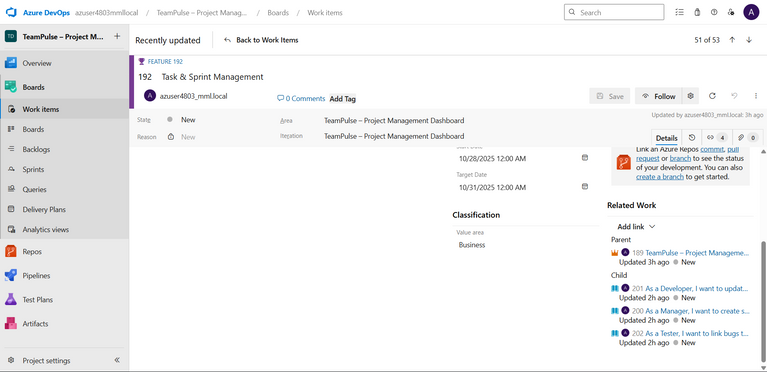
* Define sprint structure
* Document sprint creation steps
* Design sprint overview dashboard

**User Story 3.2:** As a Developer, I want to update task status so that everyone knows project progress.   
 **Tasks:**

* Document task workflow states
* Design task update interface
* Create notifications for status changes

**User Story 3.3:** As a Tester, I want to link bugs to user stories so that quality issues are traceable.   
 **Tasks:**

* Define bug linking process
* Document bug lifecycle
* Add validation for bug closure



**Feature 4: Team Chat & Notifications**

**User Story 4.1:** As a User, I want to chat with my teammates so that I can coordinate easily.   
 **Tasks:**

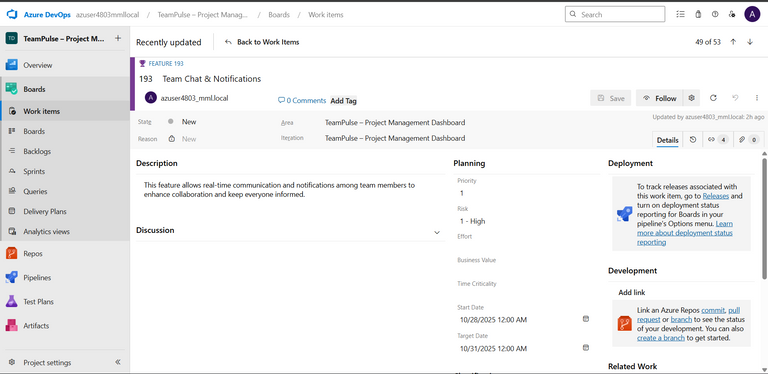
* Design chat interface
* Define message storage concept
* Document chat permissions

**User Story 4.2:** As a Manager, I want to receive notifications for critical task updates so that I can take timely actions.   
 **Tasks:**

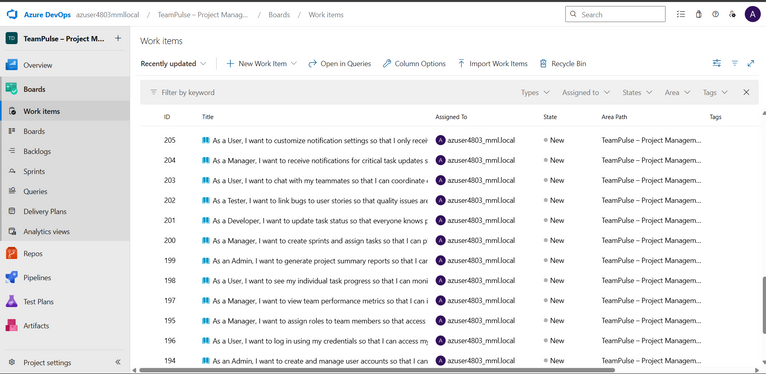
* Identify notification triggers
* Document notification workflow
* Create alert priority mapping

**User Story 4.3:** As a User, I want to customize notification settings so that I only receive relevant alerts.   
 **Tasks:**

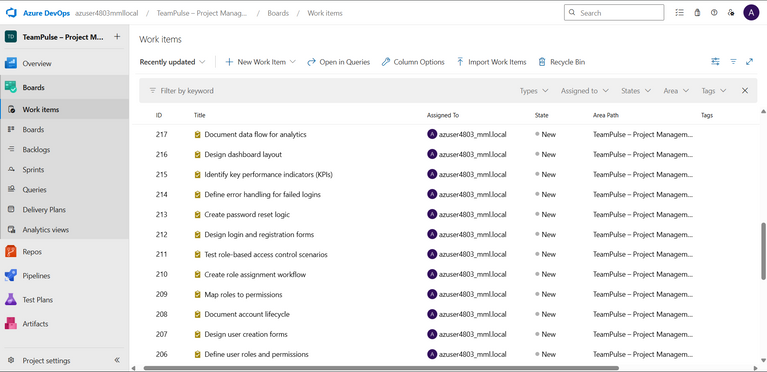
* Define notification preferences UI
* Document preference storage logic
* Test different preference combinations

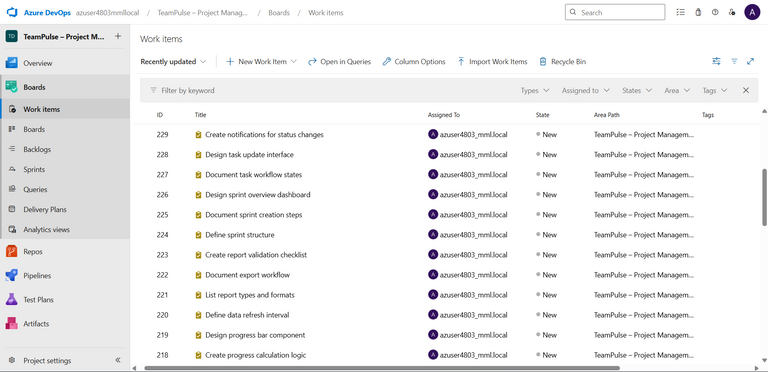


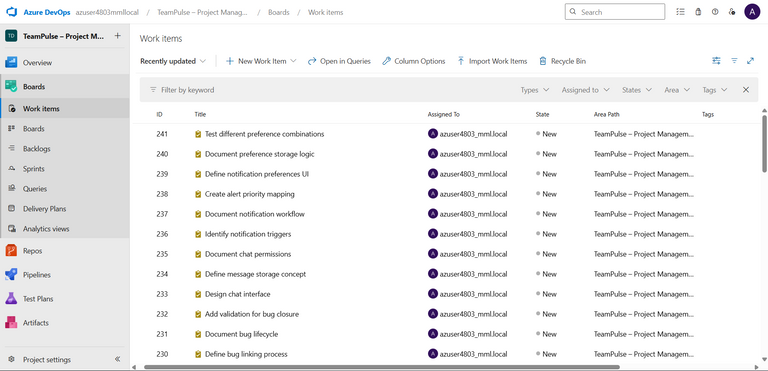
**All User Stories:**



**All Tasks:**







**Write a short explanation----**

**How Azure DevOps supports Agile methodology:**

Azure DevOps supports Agile methodology by providing tools for planning, tracking, and managing work using Agile frameworks like Scrum and Kanban. It allows teams to create backlogs, manage sprints, visualize progress through boards, and continuously deliver updates through CI/CD pipelines. This helps improve collaboration, transparency, and adaptability throughout the development cycle.

**Benefits of breaking large projects into smaller work items:**

Breaking large projects into smaller work items makes the work easier to plan, track, and deliver. It helps teams identify issues early, improve focus, and achieve faster feedback loops. Smaller tasks also enhance productivity, reduce risks, and ensure steady progress toward project goals.

**Additional Sections:**

**1. Iteration / Sprint Planning**

Sprint planning in Azure DevOps helps organize work items (User Stories and Tasks) into short, manageable development cycles. It enables teams to focus on specific deliverables while maintaining steady progress through measurable goals.

Sprint Name: Sprint 1 – Foundation Build

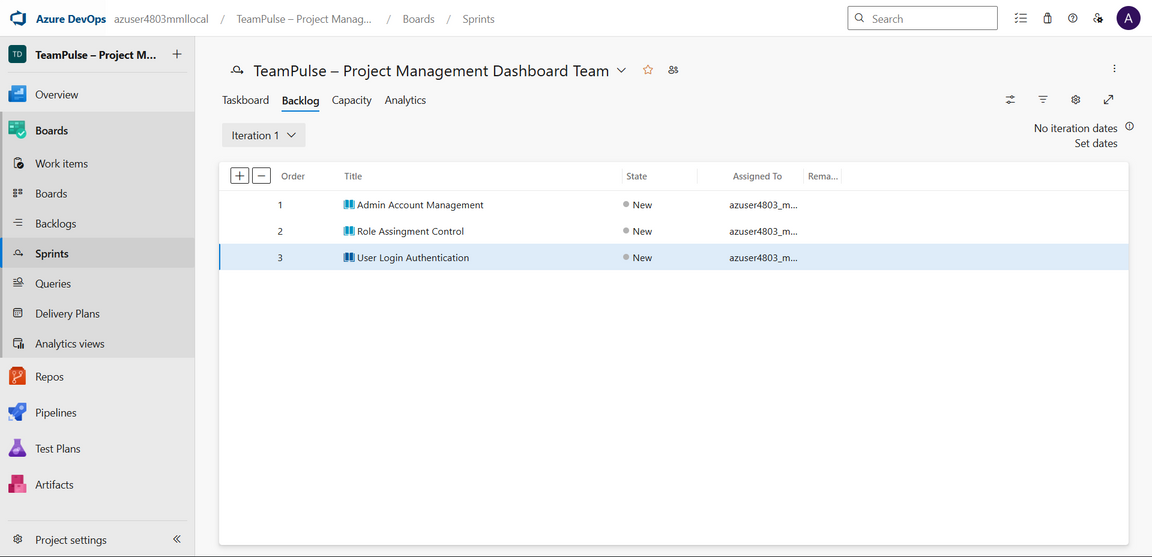
Duration: 2 Weeks

Sprint Goal: To complete setup of authentication and task management modules.

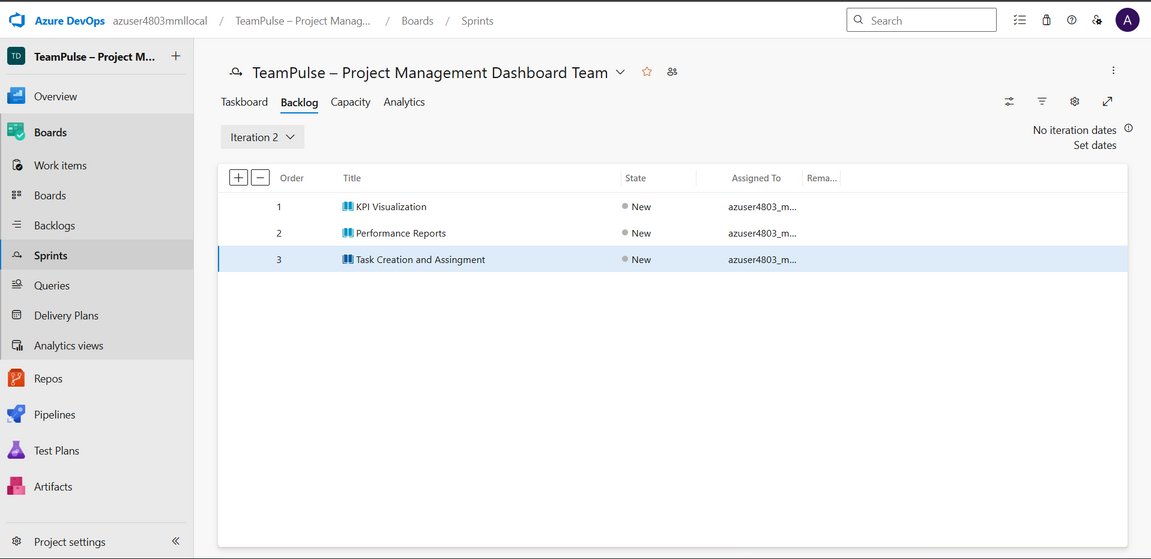
Steps to implement in Azure DevOps:   
 1. Navigate to Project Settings → Boards → Sprints.   
 2. Create Sprint 1 with defined start and end dates.   
 3. Drag selected User Stories and Tasks into Sprint 1 from the Backlog view.   
 4. Assign team members to specific tasks.   
 5. Use the Taskboard view to track progress through To Do, In Progress, and Done columns.

Sprint planning ensures structured delivery, prevents overload, and keeps the team aligned with Agile principles.

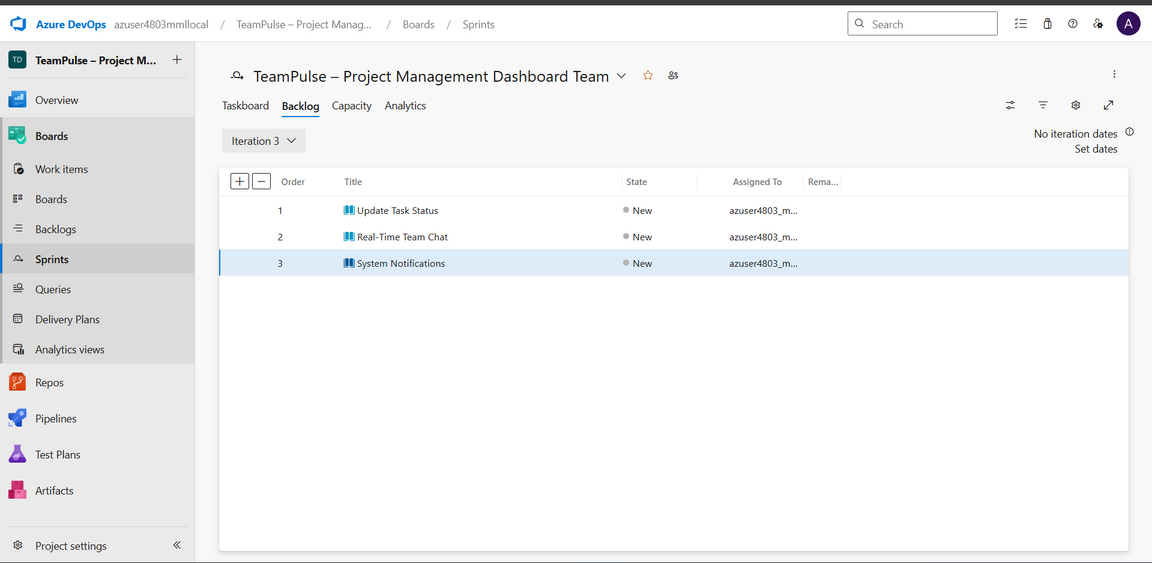
Sprint 1 – iteration 1



Sprint 1 – iteration 2



Sprint 1 – iteration 3



**2. Tags and Priorities**

Tags and priorities help classify work items by functionality and urgency, improving coordination across teams.

Example for TeamPulse:   
 • Frontend – UI elements like dashboards and chat (P2)   
 • Backend – Authentication, logic, APIs (P1)   
 • Database – Schema and data design (P1)   
 • Analytics – Reports, KPIs, charts (P3)

In Azure DevOps, tags are added directly to work items, while priorities (P1, P2, P3) indicate importance. Filtering by tags or priority ensures efficient tracking and balanced workloads.

**3. Acceptance Criteria**

Acceptance criteria define what success looks like for each User Story. They ensure a shared understanding between developers, testers, and stakeholders.

Examples for TeamPulse:   
 • Login Authentication: Must reject invalid credentials, auto-expire sessions, and redirect after login.   
 • KPI Visualization: Dashboard must display accurate sprint data and refresh periodically.   
 • Task Assignment: Tasks should link to a sprint and contain mandatory fields.   
 • Team Chat: Only project members can send messages; unread notifications must appear.

**4. Definition of Done (DoD)**

The Definition of Done ensures consistent completion standards for all work items. A task or story is considered Done when:   
 • All acceptance criteria are satisfied.   
 • Documentation is complete and peer-reviewed.   
 • Work is ready for testing or sprint closure.   
 • Dependencies are closed and linked properly.

**5. Burndown Chart**

A Burndown Chart visualizes sprint progress, showing the remaining work over time. It helps track completion trends and forecast sprint success.

Implementation in Azure DevOps:   
 • Go to Boards → Sprints → Analytics tab.   
 • Select the Burndown Chart widget.   
 • Assign all Sprint 1 items to visualize progress.   
 The chart’s X-axis represents time, and the Y-axis shows remaining tasks or story points.

This visualization helps teams identify bottlenecks early and maintain a sustainable development pace.

