Team name: Avengers

Project name: Sudo

Members:

Dmytro Senyk (Product Owner) - G00439607

Bogdan Tabalae (DevOps) - G00436707

Kostiantyn Panasenko (Quality Tester) - G00436001

Tomasz Touma (Scrum Master) - G00439630

Module name: Project Management

Assignment: Team Project Sub-Assignment 3. Scoping and Requirements Definition Brief.

Lecturer: Joseph Corr

Date: 23-Feb-2026

Table of contents

Brainstorm and Sketch of Project Requirements

At this stage of the project, our team focused on identifying the complete set of requirements for the Sudoku application.

We intentionally did not limit our thinking to the final prototype but instead brainstormed a wide range of possible features and system qualities to fully explore the potential scope of the application.

Brainstormed Functional Requirements

Functional requirements describe what the system should do. These represent features and behaviors of the Sudoku application.

Core Gameplay Features

- The system must generate a valid 9x9 Sudoku puzzle.
- The system must support multiple difficulty levels (Easy, Medium, Hard).
- The system must allow users to input numbers into cells.
- The system must validate moves according to Sudoku rules.
- The system must highlight incorrect entries.
- The system must allow undo and redo functionality.
- The system must allow puzzle reset.
- The system must include a timer.
- The system must allow users to pause and resume a game.
- The system must save game progress automatically.

Player Progress & Statistics

- The system must track completion time.
- The system must track accuracy rate.
- The system must display a statistics dashboard.
- The system must store historical performance data.
- The system must include time-based achievements.

Customization & User Experience

- The system must include dark mode.
- The system must allow customizable themes.
- The system must include sound effects toggle.
- The system must support responsive design for different screen sizes.

Learning & Assistance Features

- The system must provide hints.
- The system must explain hint logic (beginner mode).
- The system must include beginner-friendly puzzles.
- The system must include optional strategy lessons.

Brainstormed Non-Functional Requirements

Non-functional requirements describe how well the system performs and the quality attributes of the application.

Performance

- The application must load in under 2 seconds.
- Puzzle generation must occur in under 1 second.
- User input response time must be immediate.

Usability

- The interface must be intuitive for beginner users.
- The font must be clear and readable.
- Buttons must be clearly labeled.
- The UI must follow consistent design patterns.

Security

- User data must be securely stored.
- Authentication (if implemented) must protect user accounts.
- No sensitive data should be exposed.

Maintainability

- Code must follow clean coding standards.

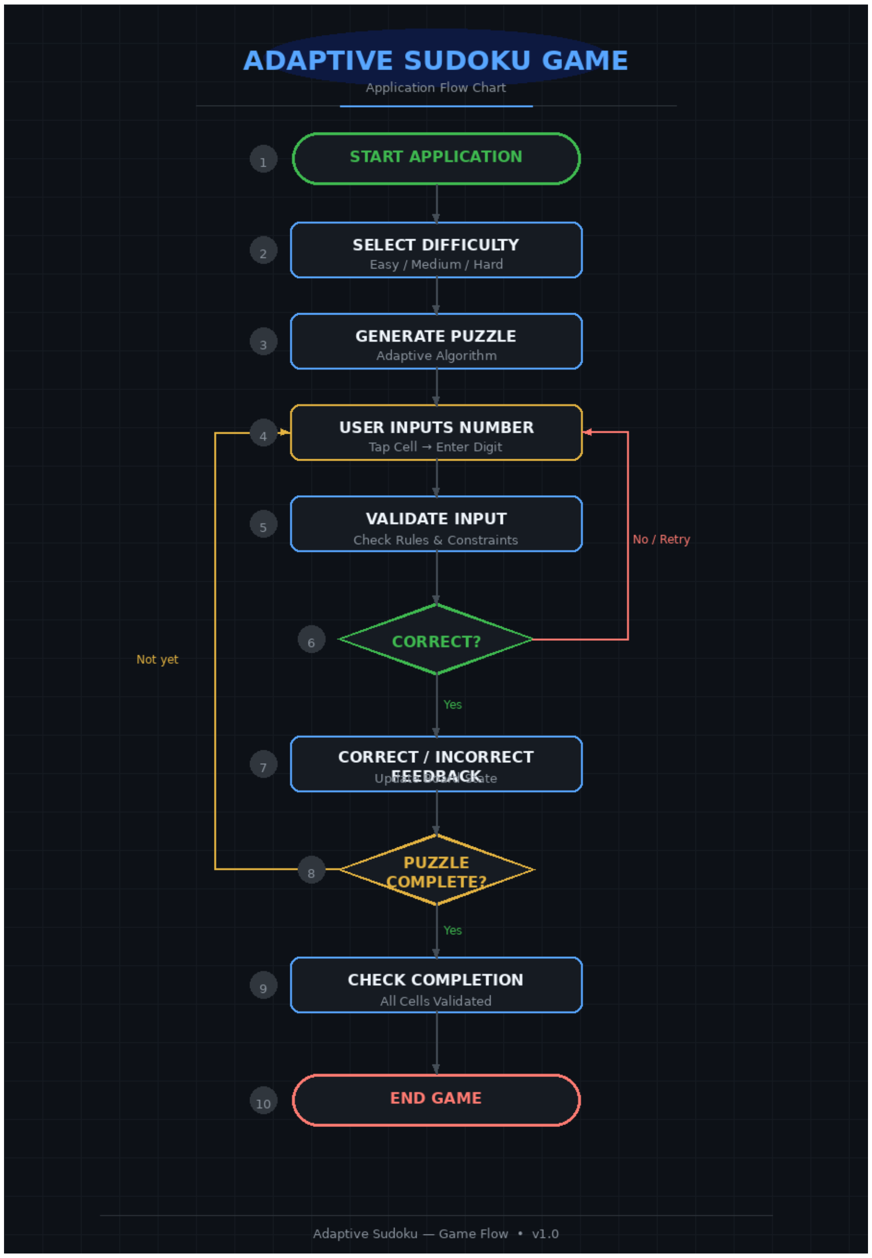
- All code must be version controlled using Git.
- Modular design must be used for scalability.

Reliability

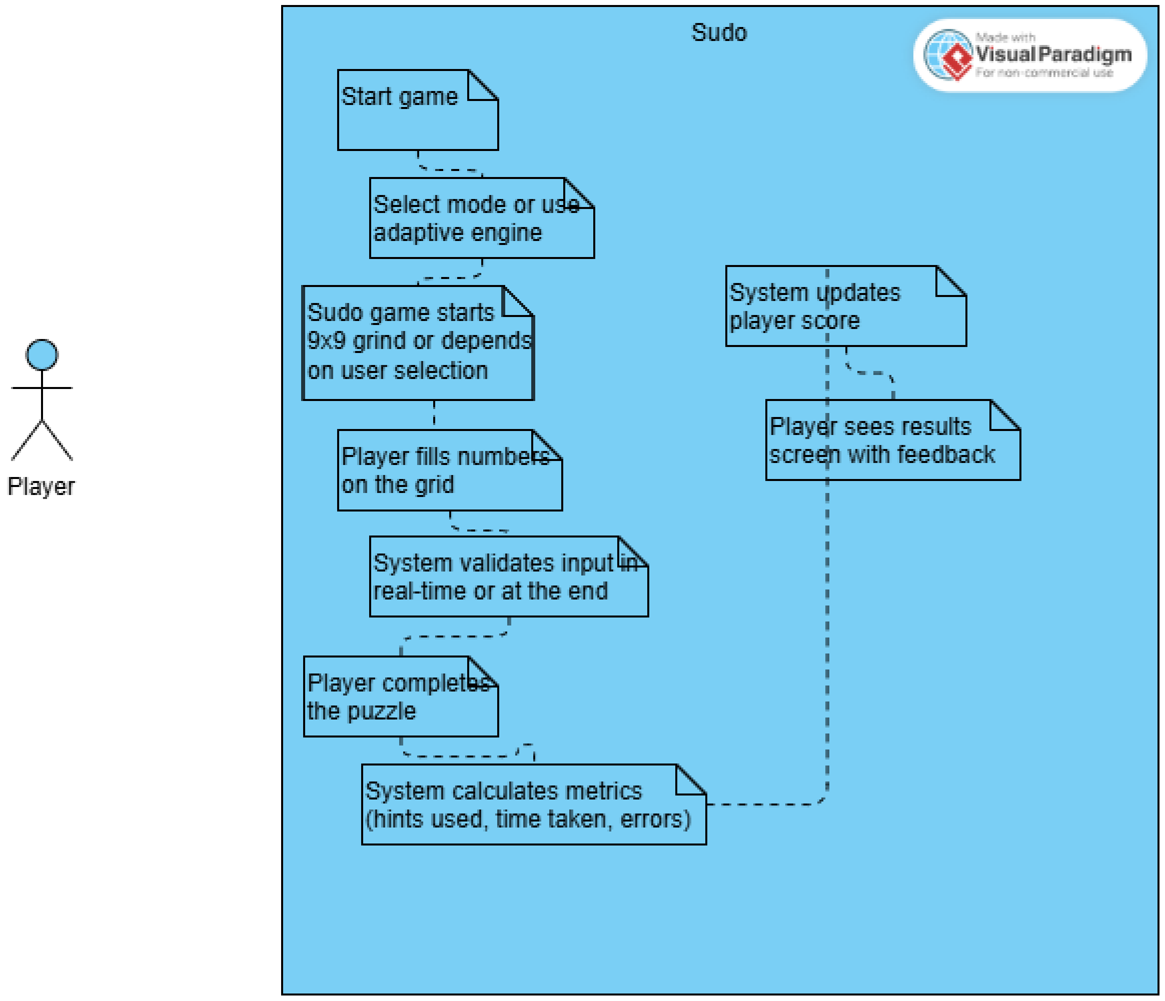
- The application must not crash during normal use.
- The system must auto-save progress.
- The application must recover gracefully from errors.

Flowchart of Core Gameplay

The following flowchart illustrates the basic gameplay process:



Use Case Overview


Sudo
Made with Visual Paradigm For non-commercial use
Start game
Select mode or use adaptive engine
Sudo game starts 9x9 grind or depends on user selection
Player fills numbers on the grid
System validates input in real-time or at the end
Player completes the puzzle
System calculates metrics (hints used, time taken, errors)
System updates player score
Player sees results screen with feedback
Player

Storyboard Sketch

The following storyboard sketches illustrate key screens of the application:

1. Home Screen

2. Game Screen
3. Settings Screen

Requirements in user story format

User Stories

The following user stories were created and documented using the 3Cs model (Card, Conversation, Confirmation). These stories are linked to Epics in Jira and sized using Planning Poker with the Fibonacci sequence.

Story 1 – Undo Move

Epic: SCRUM-19 – Core Gameplay
Story Points: 13

Card

As a user,
I want to undo moves
so that I can fix accidental entries.

Conversation

During discussion, the team agreed that:

- Undo should remove the last move only.
- Multiple undo actions should be allowed.
- Undo must not reset the full board.
- Timer behavior must remain consistent (decision: timer continues running).
- Feature complexity is high because move history must be stored.

Confirmation (Acceptance Criteria)

- An Undo button is visible during gameplay.
- Pressing Undo removes only the last entered value.
- Multiple undo operations are supported.
- The grid updates correctly after each undo.

- The application does not crash when undo is used repeatedly.

Story 2 – Choose Skill Level

Epic: SCRUM-45 – Skill Personalization
Story Points: 13

Card

As a new user,
I want to choose my starting skill level
so that I don't feel overwhelmed.

Conversation

The team discussed that:

- Skill levels will include Easy, Medium, and Hard.
- Difficulty will affect the number of pre-filled cells.
- Easy will contain more hints/pre-filled numbers.
- This impacts puzzle generation logic, making it complex.
- Difficulty must be clearly displayed during gameplay.

Confirmation (Acceptance Criteria)

- User can select difficulty before starting the game.
- Puzzle generated reflects selected difficulty.
- Difficulty level is visible during the game.
- Changing difficulty starts a new puzzle.
- No system errors occur during generation.

Story 3 – Push Notifications

Epic: SCRUM-48 – Challenges and Motivation
Story Points: 5

Card

As a user,
I want push notifications for daily puzzles
so that I don't forget to play.

Conversation

The team agreed that:

- Notifications will be optional (user can enable/disable).
- One notification per day.
- This feature may depend on platform capabilities.
- Implementation complexity is moderate.

Confirmation (Acceptance Criteria)

- User can enable or disable notifications in settings.
- A notification is sent once daily when enabled.
- Notification opens the app when clicked.
- No duplicate notifications are sent.

Story 4 – Notes Feature

Epic: SCRUM-19 – Core Gameplay
Story Points: 8

Card

As a player,
I want a notes feature
so that I can mark possible numbers in cells.

Conversation

The team discussed that:

- Notes should appear smaller inside a cell.
- User can toggle between note mode and normal input mode.
- Notes must not overwrite final answers.
- This requires additional UI logic and state management.

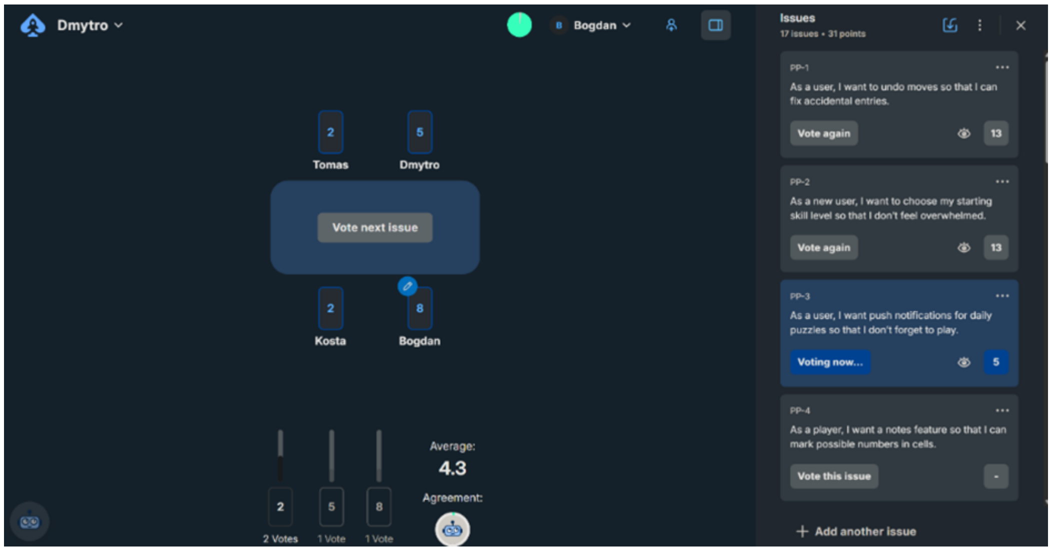
Confirmation (Acceptance Criteria)

- User can activate note mode.
- Multiple small numbers can be added in a cell.
- Notes can be removed individually.
- Notes disappear if the correct number is entered.
- System remains stable during repeated note usage.

Story Sizing Method

User stories were sized using the Planning Poker technique based on the Fibonacci sequence (1, 2, 3, 5, 8).

During a collaborative sizing session, each team member voted on story complexity, and consensus was reached through discussion.



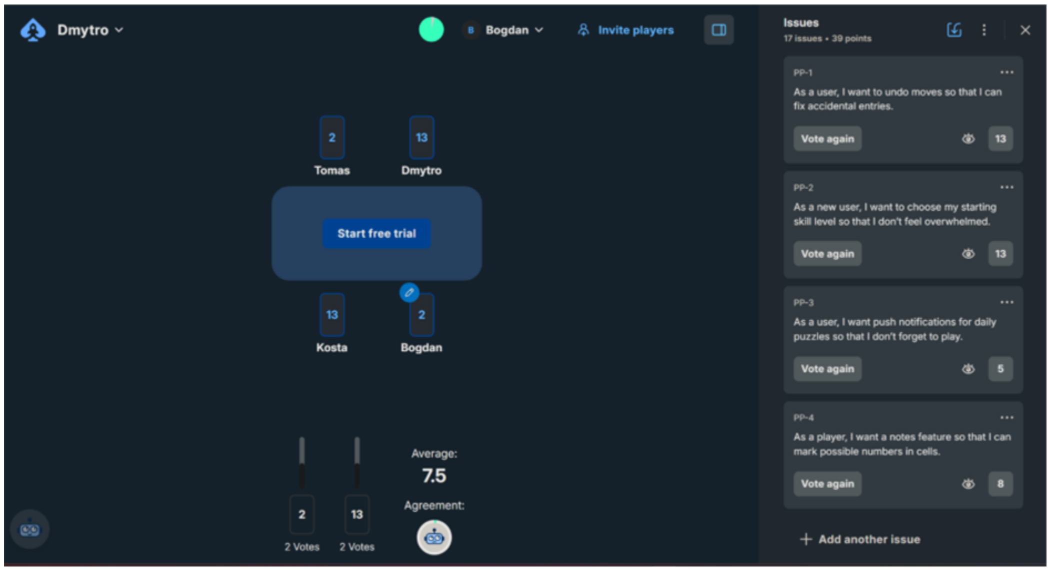
Issues
17 issues • 31 points

PP-1
As a user, I want to undo moves so that I can fix accidental entries.
Vote again 13

PP-2
As a new user, I want to choose my starting skill level so that I don't feel overwhelmed.
Vote again 13

PP-3
As a user, I want push notifications for daily puzzles so that I don't forget to play.
Voting now... 5

PP-4
As a player, I want a notes feature so that I can mark possible numbers in cells.
Vote this issue -



Issues
17 issues • 39 points

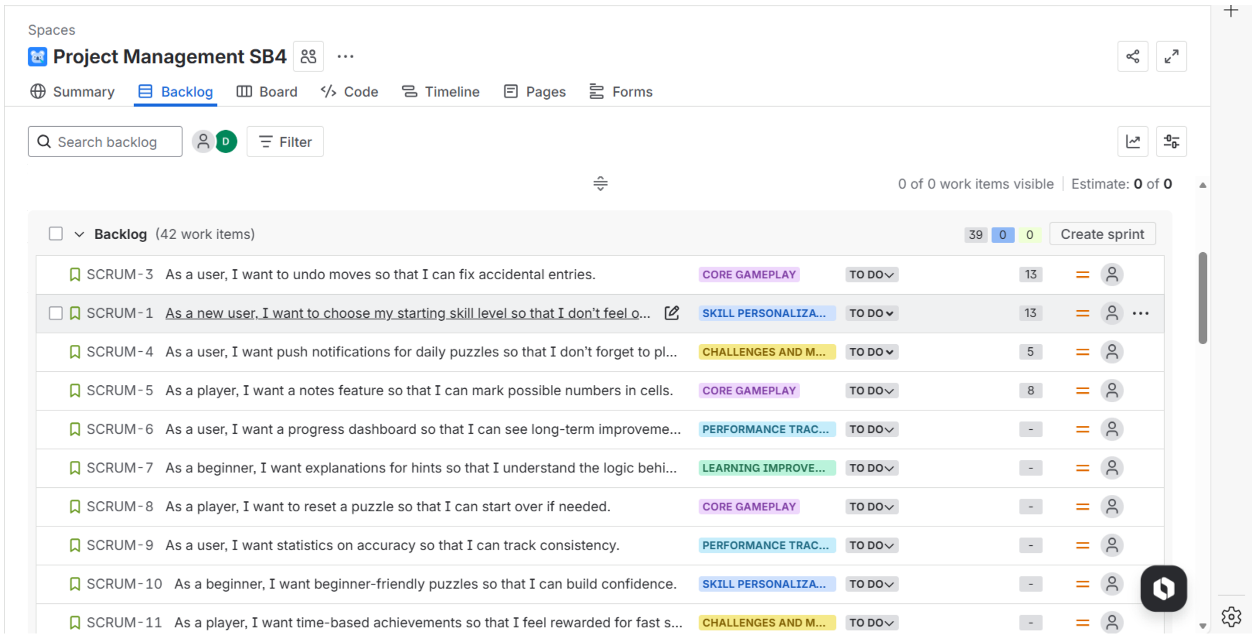
PP-1
As a user, I want to undo moves so that I can fix accidental entries.
Vote again 13

PP-2
As a new user, I want to choose my starting skill level so that I don't feel overwhelmed.
Vote again 13

PP-3
As a user, I want push notifications for daily puzzles so that I don't forget to play.
Vote again 5

PP-4
As a player, I want a notes feature so that I can mark possible numbers in cells.
Vote again 8

Grouping stories into Epics


Spaces
Project Management SB4

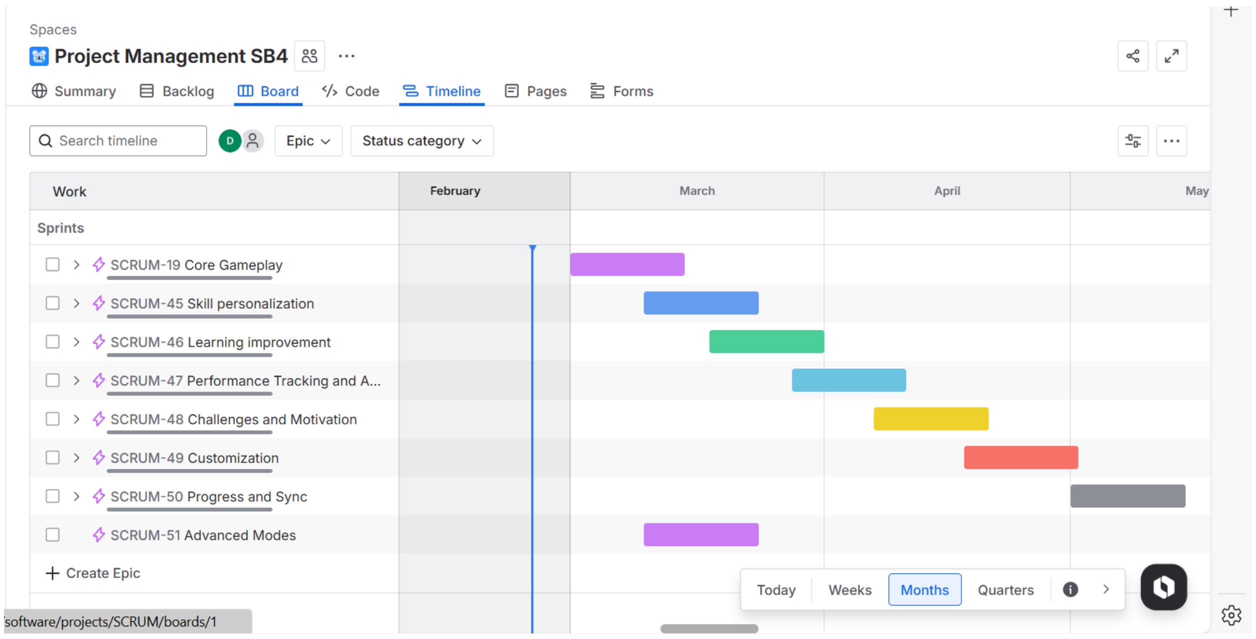
Summary | Backlog | Board | Code | Timeline | Pages | Forms

Search backlog | Filter

0 of 0 work items visible | Estimate: **0** of **0**

Backlog (42 work items) 39 0 0 Create sprint

SCRUM-3	As a user, I want to undo moves so that I can fix accidental entries.	CORE GAMEPLAY	TO DO	13
SCRUM-1	As a new user, I want to choose my starting skill level so that I don't feel o...	SKILL PERSONALIZA...	TO DO	13
SCRUM-4	As a user, I want push notifications for daily puzzles so that I don't forget to pl...	CHALLENGES AND M...	TO DO	5
SCRUM-5	As a player, I want a notes feature so that I can mark possible numbers in cells.	CORE GAMEPLAY	TO DO	8
SCRUM-6	As a user, I want a progress dashboard so that I can see long-term improveme...	PERFORMANCE TRAC...	TO DO	-
SCRUM-7	As a beginner, I want explanations for hints so that I understand the logic behi...	LEARNING IMPROVE...	TO DO	-
SCRUM-8	As a player, I want to reset a puzzle so that I can start over if needed.	CORE GAMEPLAY	TO DO	-
SCRUM-9	As a user, I want statistics on accuracy so that I can track consistency.	PERFORMANCE TRAC...	TO DO	-
SCRUM-10	As a beginner, I want beginner-friendly puzzles so that I can build confidence.	SKILL PERSONALIZA...	TO DO	-
SCRUM-11	As a player, I want time-based achievements so that I feel rewarded for fast s...	CHALLENGES AND M...	TO DO	-


Spaces
Project Management SB4

Summary | Backlog | Board | Code | Timeline | Pages | Forms

Search timeline | Epic | Status category

Work	February	March	April	May
Sprints				
SCRUM-19 Core Gameplay				
SCRUM-45 Skill personalization				
SCRUM-46 Learning improvement				
SCRUM-47 Performance Tracking and A...				
SCRUM-48 Challenges and Motivation				
SCRUM-49 Customization				
SCRUM-50 Progress and Sync				
SCRUM-51 Advanced Modes				
+ Create Epic				

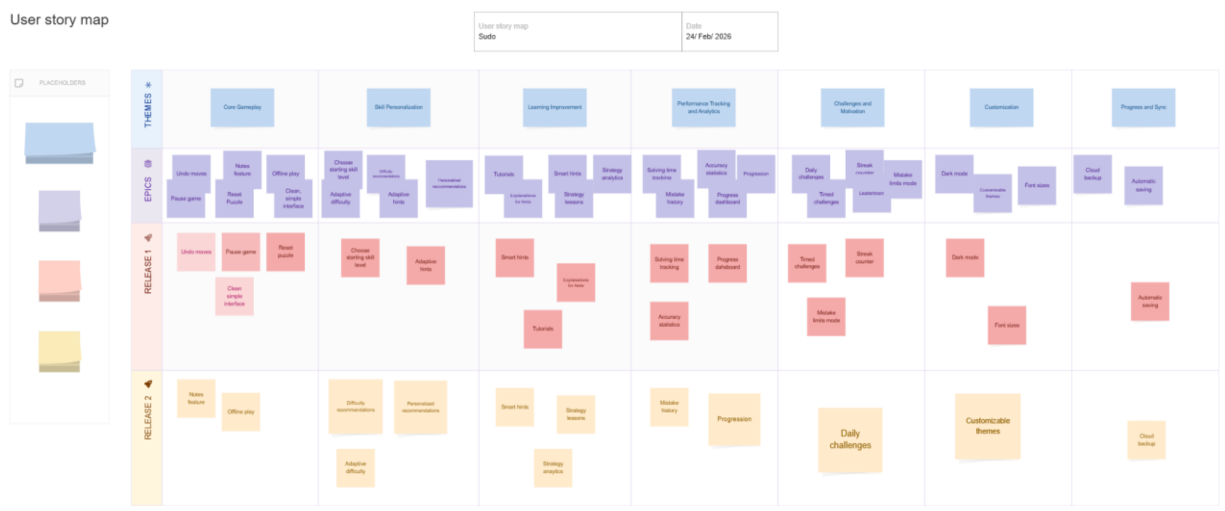
Today | Weeks | Months | Quarters

software/projects/SCRUM/boards/1

Adding user stories and epics to Jira project

Jira link: https://atu-team-g7ggluvm.atlassian.net/jira/software/projects/SCRUM/summary?atlOrigin=eyJpIjoiYTRiMWI0ZGM3YmFjNDM3YmFhOThmODVkZjQ3NmZhNmMiLCJwIjoiaiJ9

User story map



Definition of Ready (DOR)

Clear Requirements

- The feature is clearly described (what it should do)
- Acceptance criteria are defined (how we know it works)
- Any edge cases are identified (invalid input, game reset)

Technical Clarity

- It’s clear whether the work is frontend (Angular), backend (Node.js), or both
- Required API endpoints are defined (if needed)
- Data structure or format is agreed upon (Sudoku board representation)

Feasibility

- The team understands how to implement it
- Dependencies are identified (puzzle generator must exist first)
- The task is small enough to complete within one sprint or milestone

Resources Ready

- Required tools, libraries, and access are available
- Development environment is working

Definition of Done (DOD)

Functionality

- Feature works as described in the project requirements
- Frontend (Angular) correctly communicates with backend (Node.js)
- Sudoku logic works correctly aka valid board, move validation, win detection

Basic Testing

- Core functionality tested manually
- No major bugs that break gameplay
- No application crashes

Code Quality

- Code is readable and properly structured
- No obvious duplicate or unnecessary code
- No console/server errors during normal use

Build & Run

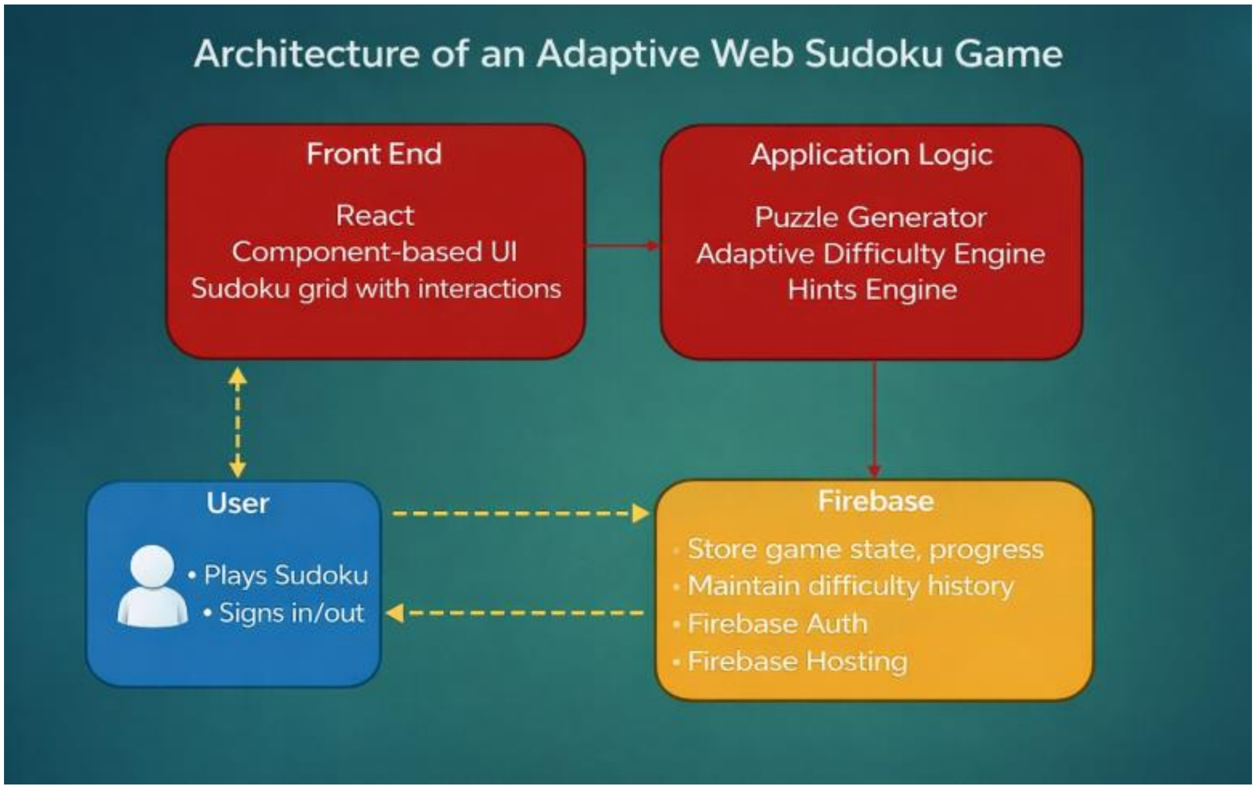
- Angular app builds successfully
- Node.js server runs without errors
- Project can be cloned and executed following README instructions

Documentation

- README includes:
    - Project description
    - Setup instructions
    - Technologies used

- Team member names

Technical Architecture


Architecture of an Adaptive Web Sudoku Game
Front End
React
Component-based UI
Sudoku grid with interactions
Application Logic
Puzzle Generator
Adaptive Difficulty Engine
Hints Engine
User
• Plays Sudoku
• Signs in/out
Firebase
• Store game state, progress
• Maintain difficulty history
• Firebase Auth
• Firebase Hosting

Development environment and tool requirement

The frontend will be built using **JavaScript** and **React**, following a component-based structure. HTML and CSS will be used for layout and styling, including the interactive Sudoku grid.

The application logic will include:

- A puzzle generator
- Adaptive difficulty engine
- Hints engine
- Game validation logic

For backend services, **Firebase** will be used. Firestore will store game state, user progress, and difficulty history. Firebase Authentication will manage user sign-in and sign-out.

Main tools:

- Visual Studio Code
- Node.js and npm
- Git and GitHub
- Firebase Console and CLI

Testing strategy, testing requirements and tools

Testing will be done continuously during development.

We will use:

- **Unit testing** (Jest) to test the puzzle generator, validation logic, difficulty system, and hints engine
- **Component testing** to check React UI elements and grid interactions
- **Integration testing** to verify connection between React and Firebase
- **Manual testing** to check usability and performance

Testing will be performed after each sprint to ensure stability.

Deployment environment and tools

The application will be deployed as a web app using **Firebase Hosting**.

Firebase Firestore will be used as the production database, and Firebase Authentication will handle user accounts.

Deployment steps:

1. Build the production version of the React app
2. Deploy using Firebase CLI
3. Test the live hosted version

The final product will be accessible through a public Firebase URL.