Entry to Tech: Assessment Criteria

Purpose of the Assessment

Assessing your technical growth

- **Foundations:** Coding, software design, file management.
- **Tools:** Different programming languages, platforms, software suites.
- **Development & Debugging:** Solving problems, troubleshooting.
- **Projects Completed:** Real-world applications and software.
- **Problem-solving:** How efficiently can you troubleshoot?

Assessing your soft skills growth

- **Communication:** Expressing ideas, listening, feedback.
- **Adaptability:** Handling change, resilience, continuous learning.

Why technical and soft skills matter

- Balancing hard and soft skills makes a well-rounded professional
- Technical expertise drives innovation
- Soft skills drive teamwork, leadership, and culture

Showcase understanding of core concepts

Demo

- Validates your learning process
- Highlights your readiness for advanced topics
- Attests to your capabilities in real-world applications

Assessment Details

Rest API

You can build any API of your choosing, but it must include the following:

- At least one GET endpoint
- Unit test at least **one** class
- Store the data in a MySQL database
- Exception handling

Rest API

- Evidence of inheritance
- Good use of HTTP Protocols
- Documentation

Be sure to read the README in its entirety to fully understand what we are looking for

Learning Outcomes

- **Design and Architect APIs:** Get to grips with the nitty-gritty of curating a top-quality API, focusing on data flow and endpoint interactions.
- **Implement Best Practices:** Showcase your adherence to Java & Spring Boot coding standards, error handling, and optimal project structure.

Learning Outcomes

- **Code Integration:** Seamlessly combine your creations with the provided skeleton codebase.
- **Exception Management:** Efficiently handle exceptions, ensuring your API remains sturdy and dependable.

Ideas

TODO List API

A file-based backend service that manages, organises, and retrieves daily

tasks

- **Endpoint:** `/tasks`
- **HTTP Method:** `GET`
- **Description:** Retrieve a list of all tasks.
- **Sample Response:** `[{ "id": 1, "task": "Buy groceries" }, { "id": 2, "task": "Finish assignment" }]`

Weather API

Weather API: A backend service that provides simulated real-time weather data by reading from predefined files

- **Endpoint:** `/weather/{cityName}`
- **HTTP Method:** `GET`
- **Description:** Get the current weather for a specific city.
- **Sample Response:** `{"city": "London", "temperature": "22°C", "condition": "Sunny"}`

Recipes API

A file-driven service that reads and serves detailed ingredients and cooking instructions

- **Endpoint:** `/recipes/{recipeName}`
- **HTTP Method:** `GET`
- **Description:** Fetch a specific recipe's details.
- **Sample Response:** `{"name": "Spaghetti Bolognese", "ingredients": ["spaghetti", "minced beef", "tomatoes"], "method": "Boil spaghetti and cook sauce..."}`

Blog or Content Manager API

A service that manages the creation, retrieval, and management of blog posts and articles using flat files

- **Endpoint:** `/articles/{articleID}`
- **HTTP Method:** `GET`
- **Description:** Retrieve a specific article based on its ID.
- **Sample Response:** `{"id": 123, "title": "The Future of Tech", "content": "Technology is constantly evolving..."}`

Power of Documentation

The Backbone of Every Project

- **Clarity:** Provides clear guidelines for users and developers.
- **Maintainability:** Simplifies future updates and changes.
- **Professionalism:** Demonstrates commitment to best practices.
- **Troubleshooting:** Helps identify and resolve issues faster.

Types of Documentation to consider

- **Naming conventions:** Helps in understanding code purpose.
- **READMEs (Markdown):** Describes project setup, purpose, and usage.
- **Listing endpoints:** Vital for APIs to provide a clear contract to users/consumers.
- **Tools:** e.g. Swagger for API documentation, Javadoc for Java
- More details can be found here

Using a project board

Why and how?

- Organise and track tasks effectively.
- Visualise project milestones and progress.
- Use platforms like GitHub Projects for setup.
- Define columns for stages: To Do, In Progress, Done.
- Regularly review and update based on progress.

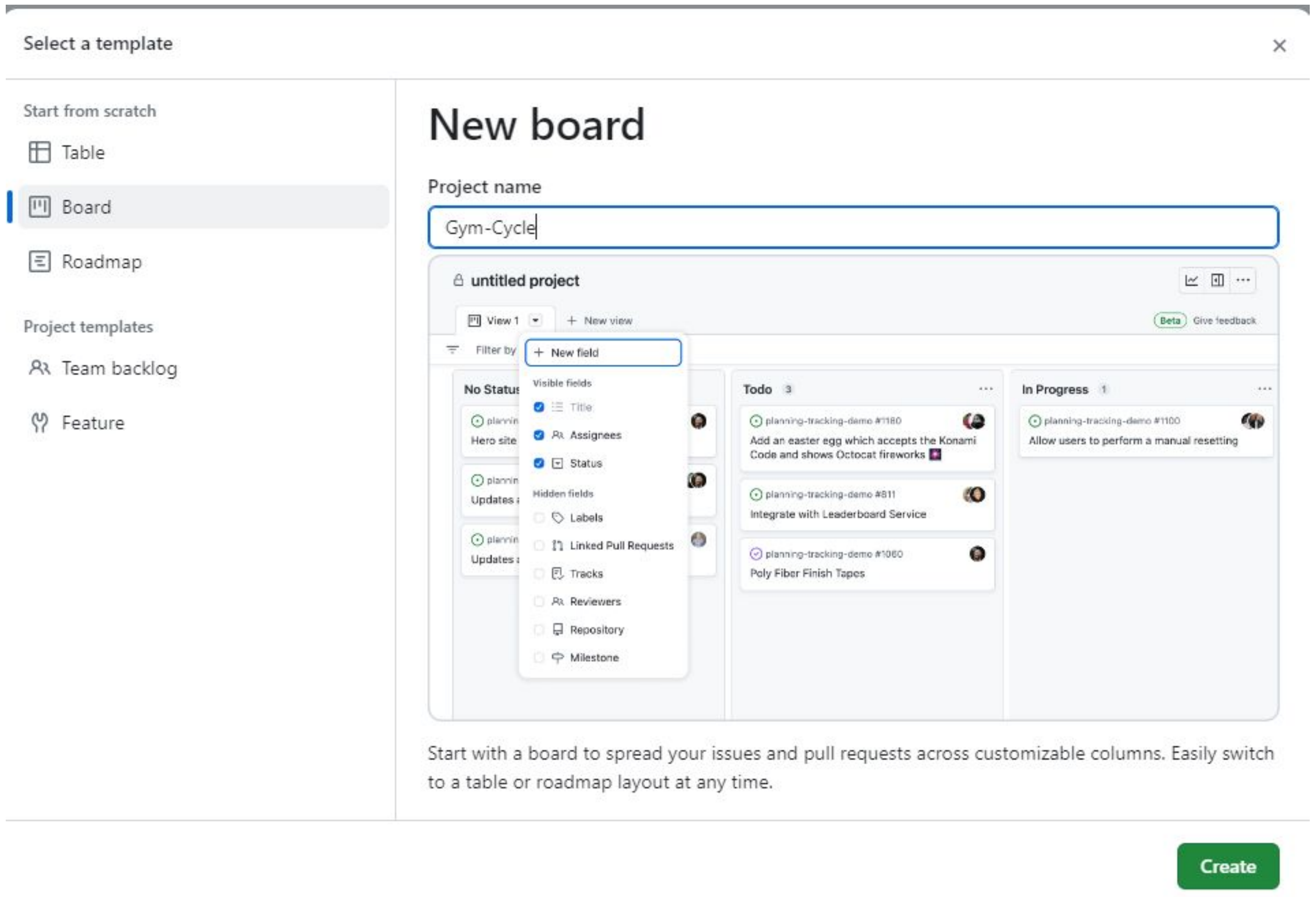
Benefits of a GitHub Project Board

- **Visibility**: Central place to see tasks, progress, and blockers.
- **Integration**: Seamlessly links to GitHub issues and pull requests.
- **Automation**: Auto-updates based on actions in the repo (e.g., closing an issue).
- **Collaboration**: Facilitates discussion on specific tasks or cards.
- **Flexibility**: Customisable columns and labels to fit project needs.

Setting up a GitHub Project Board

- Create a **New Project** in your project repository

- Useful guides to get you started:
  - Quickstart: Creating a GitHub Project
  - Best practice for GitHub projects



New board

Project name

Gym-Cycle

planning-tracking-demo #1180
Add an easter egg which accepts the Konami Code and shows Octocat fireworks

planning-tracking-demo #811
Integrate with Leaderboard Service

planning-tracking-demo #1060
Poly Fiber Finish Tapes

planning-tracking-demo #1100
Allow users to perform a manual resetting

Start with a board to spread your issues and pull requests across customizable columns. Easily switch to a table or roadmap layout at any time.

Kickstart Your Project with These Tasks

Define your primary endpoints

- **Purpose:** Understand each endpoint's core function.
- **Methods:** Identify required HTTP methods (GET, POST, etc.).
- **Data Flow:** Determine input/output data formats.
- **Error Handling:** Plan for potential errors and their responses.
- **Documentation:** Briefly describe the endpoint's role for clarity.

Design a file schema

- **Structure:** Outline data hierarchy and relationships.
- **Data Types:** Identify required fields and their formats.
- **Consistency:** Ensure uniform data entry standards.
- **Accessibility:** Plan for efficient data retrieval and updates.
- **Documentation:** Describe file layout and data conventions.

Integrate essential libraries

- **Selection:** Choose libraries based on project needs.
- **Compatibility:** Ensure libraries work seamlessly together.
- **Efficiency:** Opt for lightweight options to boost performance.
- **Documentation:** Consult library docs for proper integration.
- **Versioning:** Ensure up-to-date and stable library versions.

Decide on initial user stories or features

- **User-Centric:** Think from the user's perspective.
- **Prioritisation:** Focus on core functionalities first.
- **Clarity:** Clearly define the expected outcome of each story.
- **Feasibility:** Ensure stories are achievable within the project scope.
- **Iteration:** Remember, stories can evolve based on feedback.

Break tasks into smaller chunks

This will help you manage your work more efficiently

- **Manageability:** Easier to tackle and debug.
- **Clarity:** Clearer objectives for each chunk.
- **Progress Tracking:** Monitor advancements step by step.
- **Flexibility:** Adjust or pivot without overhauling everything.
- **Momentum:** Achieving little wins boosts morale and motivation.

Practice Task

GitHub Project Board Task

Set up a GitHub Project Board for a mock project (use one of the ideas previously mentioned if you can't think of one)

- **Goal**: Add 5 initial cards/tickets to your board
- **Bonus**: Attach a basic README to your mock project repository

You’ve got this!

You’ve got this!

- This assessment is a reflection of your journey and growth
- Embrace challenges and keep learning
- Good luck, and enjoy the process!