🔖 User Stories & Product Requirements

User Stories

Epic No. 1	Stories	Story No.	Priority
Create, delete, rearrange, regenerate parsons problems.	• As a user I want to have a tool that can generate parsons problems. • Problems relate to data analytics. • Problems are AI generated.	1.0	**High**
	• As a user, I want to select the topic and context of a problem before generating.	1.1	**High**
	• As a user I want to be able to interact with the generated problem through a drag and drop interface.	1.2	**High**
	• As a user I want to have the ability to re-generate a problem with a different context, topic or with the same context and topic.	1.3	**Medium**
	• As a user, I should be able to save the problems that I solved, or generated. • I also then want to delete selected saved problems.	1.4	**Low**

Epic No. 2	Stories	Story No.	Priority
Generate, save, check feedback for parsons problems	• As a student I want to be able to submit problems when finished.	2.0	**High**

	• As a user I want to receive correct or incorrect feedback for a submitted problem.	2.1	**High**
	• As a user I want to have code error response for a submitted parsons problem in the same format as a compiler or IDE error codes.	2.2	**Low**
	• As a user I want to be able to receive personal statistics.	2.3	**High**
	• As a user I want to check for problem solutions after a number of incorrect attempts.	2.4	**Low**

Product Requirements

The following requirements correspond to the Epic No. and Story No. shown in the above tables

Epic No. 1 & Story No.	Functional Requirements	Potential challenges and changes	Priority
Story No. 1.0	• Feature to connect to the Gemini api and have a category and context selection before generating prompt. • Gemini's response must be formatted correctly, and split into lines. Each line order should then be randomized and displayed to the user. • Gemini must also generate some problem description with the correct context selected by the user.	• AI model might change depending on cost considerations. • The complexity and types of problems are limited to the model. • Ambiguous problem descriptions may occur due to the Gemini hallucinating.	**High**
Story No. 1.1	• The web interface requires a page containing the	• code length might be restricted to ensure	**High**

	categories, and the contexts that users can select from. • After selecting the contexts categories, users should see their generated problems in a workspace page.	that problems are not exhaustively long.	
Story No. 1.2	• Gemini response should be able to be dragged from question allocation to the answer location. • Users should be able to re-order blocks in both question and answer location.	• drag and drop may be limited as a pre-existing drag and drop library will be used.	**High**
Story No. 1.3	• Regeneration button that has two options, regenerate current prompt or regenerate prompt with new context and category.		**Medium**
Story No. 1.4	• Save option for a generated problem on the workspace page. • Alternate page to display saved problems and delete saved problems.	• Saved option may be limited to registered users.	**Low**

Epic No. 2 & Story No.	Functional Requirements	Potential challenges and changes	Priority
Story No. 2.0	• Have a submit button.		**High**
Story No. 2.1	• After failed submissions users should receive AI generated feedback.		**High**
Story No. 2.2	• Submissions should return a compiler-like error code in the workspace page.	• Accurate, well formatted error codes may be challenging as it requires an	**Low**

		online python interpreter. • This requirement may change depending on the client's request. The bare minimum functionality should be Boolean checks eg. correct or incorrect.	
Story No. 2.3	• Have a designated page for user related statistics, profile and basic analytics.	• Graphs and other statistics visualisations may be added depending on the client's future request.	**High**
Story No. 2.4	• After a number of attempts, users should be able to check their solutions on a solutions page for the generated problem.		**Low**