User Story and Backlog

The user story page has been reuploaded in response to feedback requesting a major rework to the user story.

Size Estimation

The size estimation provides an estimate of the amount of time required to complete the corresponding User Story (US) or the subtask. Each task is assigned a certain amount of *story points* (s.p.). The smallest amount of story points possible is 0.5 s.p. One story point is equivalent to one day's worth of work from one developer. Likewise, 5 s.p. would mean that it would require one developer five days to complete a task. However, it should be noted that since the developers of this project are students, they may have other subject commitments and may need extra time to familiarise themselves with the technology. Hence, each story point may represent 1.5 - 2 days' worth of effort for each developer in this team.

Based on the number of s.p. required, all US and subtasks are classified into one of five categories:

- Very small: less than 1 s.p.
- Small: 1 - 2 s.p.
- Medium: 3 - 5 s.p.
- Large: 6 - 9 s.p.
- Very Large: 10+ s.p.

MoSCoW Priority

MoSCoW Priority rating assigns a level of importance to each of the subtasks. The more important it is, the more critical that it should be completed. In order of descending importance, there are:

- Must have: these are tasks that must be completed, otherwise the product will lose most/all of its value and fail to meet its goals.
- Should have: these are tasks that if completed, will bring significant convenience to the users.
- Could have: these tasks, when completed will give a small amount of benefit to the users. However, the application will function fine without them.
- Will not have: these features bring minimal value to the overall product and are hence considered unworthy of time investment. They will not be developed unless there are ample time.

Epic ID	Epic	User Story ID	As	I want to	So that	US Size Estimation	Subtask ID	Subtask	MoSCoW Prio.	Priority Justification	Size Esti.	Size Justification	Acceptance Criteria
EP-1	Construct an answer	US-1.1	Gloria Baldwin (student)	Have a collection of logical connectors	I can connect the statements and form my solution.	Medium 4 s.p.	T-1.1.1	Construct the shape of the connector: a space for statements on the left and right, with a connector word shown in the middle.	Must have	Without the ability for students to connect their statements, the students will not be able to construct their solution to the problem. This subtask is the starting point for US-1.	Small 1 s.p.	This is just some basic framework that all other features will extend from; very barebones and hence is only estimated to be a small task.	A basic (can be empty) framework, imitating the structure already built in Python can be displayed.
							T-1.1.2	Display a link word /icon for each connector	Must have	Without the link word displayed, the student will not know how they are linking the statements together.	Small 1 s.p.	The centre part of the connector is already built; we simply have to define the words that go in there for each unique connector.	The central space of the connector should contain a customizable word or image.
							T-1.1.3	Change the form of the connector	Must have	Sometimes, specific link words are required to make the sentences flow. In addition, the client has requested T-1.2.3 to be of most priority as it also indirectly shows the understanding that the student has.	Small 2 s.p.	Changing to most of the other keywords are quite simple as they are also one word/phrase between two statements. However, some of the desired statements are in a different form (e.g., if [A] then [B]). These statements could be difficult to implement.	Upon some form of interaction (likely clicking and selecting from a dropdown menu), the connecting word changes to the selected one with the correct syntax.
		US-1.2	Henry Paulson (assessor)	Have my predefined statements be shown to the student	students can use them to construct their answers and I can assess their learning.	Large 6 s.p.	T-1.2.1	Display all the statements defined in the question file.	Must have	Without displaying the statements, the students will not know what statements they have and therefore cannot construct an answer.	Medium 4 s.p.	There are three parts in this subtask. We will need to be able to read the JSON content. We will need to build the area to display the statements, and we need to change the information in the JSON file into actual statements.	Upon opening a question file, all the statements that are available to use should be displayed to the user.

						T-1.2.2	Display different types of statements, based on the type defined in the JSON file.	Must have	Some statements are required while others are optional. Without differentiating features between the statements, the student might not construct a proper solution and the assessor cannot accurately assess the student's learning.	Small 1 s.p.	The type of statement is contained in the JSON file. We need to ensure that the style of the statements displayed matches the current Python implementation.	If there are different types of statements defined in the file, they should be displayed with distinct features between them.
						T-1.2.3	Allow only the set amount of statements to be shown and used.	Must have	The number of statements that the students used can also be an indicator of the student's understanding of the problem. Hence, allowing the student to only use a certain amount of statements is required.	Small 1 s.p.	The JSON file contains the number of statements that should be displayed.	There should only be as many statements as there are defined on the whole page; once one statement is used, it should be unavailable in the display area and become one part of the answer.
		US-1.3	Gloria Baldwin (student)	drag statements and connectors on top of each other	I can construct an answer using statements and connectors.	Very Large 10 s.p.	T-1.3.1	The connector should be able to be dragged onto the workspace.	Must have	If the connectors cannot be dragged onto the workspace, the students cannot build their answers to the question.	Medium 3 s.p.	There are two parts to this subtask. Firstly, the connector itself will need to be dragged around. Secondly, the workspace will need to be able to accept a connector.	Upon attempting to drag a connector to a workspace, the connector should be correctly shown in the workspace without consuming the original connector.
							T-1.3.2	The statements should be able to be dragged onto the connector's left and right sides.	Must have	The student cannot connect two different statements if the connector cannot accept them.	Medium 3 s.p.	To be able to drag statements on top of connectors, we need to allow statements to be dragged and the connectors to accept the dragged statement. Afterwards, the connector will need to display the statement stored.	1. When dragging a statement onto a connector in the workspace's left or right component that is **empty**, the corresponding component should successfully display that statement. 2. When dragging a statement onto a connector in the workspace's left or right component that is **not empty**, a replacement confirmation message should display. 3. If confirmed, the old statement should be **replaced** by the new statement, and the old statement should become available to use again.

							T-1.3.3	A connector should be able to be dragged onto another connector's left and right sides.	Must have	To build a more extended answer, multiple connectors and statements will be required. Hence, one part of the built sentence (i.e., connector) will need to be able to connect via another connector.	Medium 4 s.p.	When dragging a connector on top of another connector, the process will need to transfer a large amount of information: the dragged connector's statements and the connector's own information. In addition, further on in the answer, a dragged connector may contain more connectors within it. Hence, this could be a very complicated issue. Therefore, we are assigning it to be a relatively large task.	1. When dragging a connector onto a connector in the workspace's left or right component that is **empty**, the corresponding component should successfully display that connector, including all the statements in the dragged connector. 2. When dragging a connector onto a connector in the workspace's left or right component that is **not empty**, a replacement confirmation message should display. 3. If confirmed, the old connector should be **replaced** by the new connector, and the old connector's child statements should become available to use again.
		US-1.4	Gloria Baldwin (student)	see the answer I've constructed	I can make sure my solution is what I intended to write.	Medium 4 s.p.	T-1.4.1	Each connector should be able to output the two statements and the link words in one phrase.	Must have	If a connector cannot output its corresponding content, then we cannot obtain the entire solution in words.	Small 2 s.p.	Each connector needs to be able to output the content of its left children, itself and the right children. However, some of the more complex connecting statements may make this part difficult, hence we are giving an extra s.p. to this subtask.	*Development testing phase only - DISABLE IN DEPLOYMENT:* when performing a specific action (e.g. clicking an icon on a connector), the application should show somewhere the worded output of this connector, which is composed of the two statements (or placeholders if they are empty), linked by the corresponding linking words.

							T-1.4.2	If a connector contains more connectors, it should successfully output the worded phrase from all connectors.	Must have	Not only do we require the most basic connectors containing two statements to be able to output their content, but we also need those that contain another connector to produce their content to successfully output the entire solution in words.	Small 1 s.p.	If each "basic" connector can already output its content, a connector of connectors should be able to also easily output its content, so we believe this is a relatively small subtask.	*Development testing phase only - DISABLE IN DEPLOYMENT:* when performing a specific action (e.g. clicking an icon on a connector), the application should show somewhere the worded output of this connector, even if there are more connectors as its children, linked by the corresponding linking words and have the correct grammar.
							T-1.4.3	The corresponding area should display the solution in words every time it is updated.	Must have	If we do not update this statement every time it has been updated, the student will not get immediate feedback on their construction and it may require excessive work to correct their mistake.	Small 1 s.p.	We believe this task is small as it is an update every time the content changes, which can be easily implemented through events in JavaScript.	Every time an element in the workspace changes (e.g. added connectors, removed statements etc.), the corresponding text output shown at the bottom of the screen should be updated to reflect the change.
EP-2	Question selection, display and submission	US-2.1	Gloria Baldwin (student)	choose the question I want to work on	I can decide on which question to answer first.	Medium 3 s.p.	T-2.1.1	The application should first log into the server, getting a unique ID which it will keep.	Must have	Updated: The protocol has been implemented on the backend and is required for further access, making it a must have. (Previously: The backend protocol defines that each client should have a unique ID. However, this is not a priority as it does not affect the question-building functionality of the program.)	Very Small 0.5 s.p.	A login function is common and the current protocols implemented in the backend server are well defined. It should be easy to complete this subtask.	Upon the launch of the application, it should send the client's predetermined ID in a POST request detailed in the documentation and successfully obtain the persistent secret key used for this session.
							T-2.1.2	The application should be able to request the list of questions from the server.	Must have	The protocol has defined that the backend server will provide the application with a list of questions. The application itself does not store information on the list of questions before this.	Small 1 s.p.	While this function is also well documented, as this is the first must-have feature of this category, it is likely that it will take more time to experiment with AJAX calls. Therefore, we would like to assign this subtask more time in case of unforeseen circumstances.	The client should be able to successfully obtain the available questions (ExNets) on the server via a GET request.
							T-2.1.3	Display the list of questions that students can select.	Must have	In order for the student to choose the question they would like to work on, the application must be able to show the student the list of questions.	Very Small 0.5 s.p.	This subtask involves showing the user a list of questions. We believe this task should be quite straightforward.	Upon performing a certain action (e.g. pressing a button), a list of questions corresponding to the list of questions received in T-2.1.2 should be shown to the student.
							T-2.1.4	Request information on the question the user has selected.	Must have	We need to request the question the student has selected in order to display it, as the application does not store information on the questions.	Very Small 0.5 s.p.	Now that we have previous experience requesting content from the server as outlined in T-2.1.2, further similar requests should be easy to implement.	Upon the selection of a question by the student, the client should send a GET request for the selected question and successfully receive information about the selected question.

							T-2.1.5	Display the student's selected question.	Must have	We need to show the student the question they selected so that they can work on the question.	Small 1.5 s.p.	This subtask involves us displaying a completely new question, wiping out all the content currently present on the screen, which may take a little time to work on.	Upon receiving a successful response from the server, the application should display the contents of the response, in the format that is consistent with the application.
		US-2.2	Gloria Baldwin (student) Henry Paulson (assessor)	See the text of the question	I know what I am responding to. I can test the students' understanding.	Small 1 s.p.	T-2.2.1	Display the question prompt in the corresponding area.	Must have	If the student cannot see the question, they will not know what they are responding to. The assessor cannot get a true understanding of the students' comprehension if they do not know what the question is.	Small 1 s.p.	As defined in the backend server protocol, the question is written in HTML, hence it should be simple to display it on a web page.	Upon loading a question, the question's query section should be shown in the corresponding area.
		US-2.3	Gloria Baldwin (student)	Submit my answer	I can complete my assessment.	Medium 3 s.p.	T-2.3.1	Convert the student's answer into a JSON format consistent with the backend implementation.	Must have	In order for the answer to be correctly received in the backend, it must follow the currently defined format as documented.	Small 2 s.p.	The documented backend format includes a lot of information beyond the content, but also information such as the position of the statement in the parent component. Hence, it may take a bit of time to extract all the features. Furthermore, the statements that involves a choice from the student have a unique format which may take some time to implement.	*Development testing phase only - DISABLE IN DEPLOYMENT:* Upon performing a certain action, the application should be able to convert the current answer into a format that is consistent with the backend implementation and is accessible to the user. This can be in the form of a pop-up containing the content or in a separate file available to be viewed later.
							T-2.3.2	Send the JSON answer to the backend server.	Must have	The server will not know if the student has submitted until the information is delivered to the server.	Small 1 s.p.	Sending information to the backend server should be relatively straightforward, so we are assigning it as a small task.	The JSON data formed in T-2.3.1 should be able to be sent and received by the backend server via a POST request.
EP-3	User Experience	US-3.1	Gloria Baldwin (student)	be able to keep my workspace tidy	I can easily see the answer I have constructed.	Very Large 15 s.p. ~~Large~~ ~~7 s.p.~~	T-3.1.1	Toggle the connector to display statements horizontally or vertically.	Should have	If the connector can already be collapsed into a single sentence, then toggling the connectors to be horizontal or vertical becomes less important.	Medium 3 s.p.	Displaying the content horizontally versus vertically also heavily modifies the original shape built. More time may be needed to make sure that the toggle works well.	Upon performing a certain action (e.g., clicking an icon), the statement that was displayed horizontally (left-mid-right) should be displayed vertically (top-mid-bottom), with the same statements and link words.
							T-3.1.2	Collapse the contents of the connector into a single sentence.	Must have	The student would like to be convenient in viewing their answers; hence, toggling such that the workspace is tiny is a must.	Small 2 s.p.	Converting the entire connector to a single sentence requires access to both of the statements stored within, in addition to the connector word in the middle. As connectors can be contained in connectors, this recursion could make the feature tricky so we are assigning it a bit more time.	Upon performing a certain action, the statement can be converted to a simple phrase that encapsulates all the information contained within it, similar to T-1.4.2.

						T-3.1.3	Updated: Delete connectors and drag statements onto workspace. ~~Delete statements and connectors when the student has made a mistake.~~	Must have	If there is no way to delete statements, if the students have made a mistake, their space will become extremely cluttered and drastically impact the tidiness of the workspace.	Small 2 s.p.	While deleting connectors may be simple as they are infinite, deleting statements needs more work as we also need to restore the deleted statement to the list of statements the students can select from, hence we are assigning this a few more story points.	Upon dragging a statement or connector in the workspace to the bin icon, the corresponding component should be removed from the workspace. Any statement removed should be available to be used again. Any connector including statements should also have their statements restored.
						T-3.1.4 (new)	Move the root connector around.	Must have	A root connect should be able to be moved around the workspace to a student's desired location.	Small 2 s.p.	Moving around an item in a space that is already droppable with more items may cause unexpected behaviour, so we are assigning more time for this task.	TBA
						T-3.1.5 (new)	Move items around in the workspace.	Must have	Moving an item around in the workspace assists in the user's logical thinking; it is also shown in the student demonstrations that many student would appreciate this function.	Medium 4 s.p.	Moving items around will not only encompass the requirements in T-3.1.4, but also allowing statements to be dragged onto the workspace, and dragging more items from current connectors out onto empty space. This task may require significant effort to achieve.	TBA
						T-3.1.6 (new)	Add in multiple root statements.	Must have	Allowing multiple root statements allows for the idea of paragraphs to exist within the answer, which assists with thinking.	Small 1 s.p.	Given that T-3.1.4 and T-3.1.5 is complete, allowing multiple root statement should require very little to the change.	TBA
		US-3.2	Gloria Baldwin (student)	Resize different sections of the display	I can focus on the important parts of the page.	Small 2 s.p.	T-3.2.1	Resize different parts of the screen by dragging the edges.	Should have	While it will most certainly help the student focus on the right content, the student can also collapse part of the answer so that it only consumes very little space, making this feature more of a nice-to-have than must-have.	Small 1 s.p.	Dragging to resize different components should be rather straightforward as it is different CSS components that we need to modify, hence we are assigning this to be a small task.	When dragging the edges between the different sections of the page, the connected sections should resize appropriately based on the drag movement.
							T-3.2.2	Zoom in and out on the workspace.	Could have	Zooming in and out may help some students in their responses. However, if we have implemented T-3.1.2, T-3.1.1 and T-3.2.1, it is unlikely that this feature will be very useful to most students. Hence, we are assigning this to be a low priority subtask.	Small 1 s.p.	While zooming in and out of the whole page may be easy, zooming in and out on a specific part of the page might be more difficult, hence one full story point may be required.	Upon performing a certain action (e.g., dragging a zoom slider), the workspace section should be zoomed in or out depending on the action taken.
							T-3.2.3 (new)	Make one part of the area full screen.	Must have	Focusing on a specific part of the area allows the student to concentrate on the content in that area. Research has also shown students would prefer such a feature being available.	Small 2 s.p.	We need to be able to open a specific part of the screen, while still preserving the other area's content. Furthermore, we may need to allow the user to edit in this focus view, which should be reflected when exiting this view. Hence, we are assigning 2 story points.	TBA

US-3.3	Gloria Baldwin (student)	have some tutorials within the application	I can learn how to use the application.	Medium 4 s.p.	T-3.3.1	Create a basic webpage with sections outlined in the high fidelity prototype.	Must have	The basic construction of the webpage is a base for subsequent development. Thus, it has a high priority.	Small 1 s.p.	This item requires dividing the sections of the page and displaying the section titles, so only need to list a simple structure by using HTML. Thus, it would be a small task.	Given: 1. I navigate to the webpage. When: 2. I want to know the meaning of each section in the webpage. Then: 3. The clear layout and meaningful section name are displayed on the webpage.
					T-3.3.2	Display tooltips throughout the application to guide users in utilizing its various features, with particular emphasis on connectors. Provide clear instructions for dragging statements onto connectors, removing statements, collapsing connectors, and accessing other implemented functionalities.	Must have	Students may forget how to interact with parts of the application under the high-stress assessment situation. Hence, there might be some easy-to-use, visible hints on how the application is used. Furthermore, the students are being assessed on their academic performance, not how well they can use the application, so this process needs to be as simple as possible and with as much information as possible available to the student on-the-fly.	Small 1 s.p.	There are plenty of guides on how to implement tooltips on the internet, and the original version of the program contains what should be written in each tooltip, so this should be quite straightforward.	1. When hovering over the side components of a connector when it's **empty**, it should tell the user that it can accept items by having them dragged onto it. 2. Upon hovering over the side components of a connector when it's **not empty**, it should tell the user how to remove said content. 3. Upon hovering over a statement, it should describe the type of statement and whether it should be used (e.g. root statements are expected in an answer). 4. Upon hovering over a connector on the right, it should tell the user how it can be used to connect statements. 5. Upon hovering over the rubbish bin, it should tell the user how to use the bin.
					T-3.3.3	Have a tutorial on how to use the program.	Could have	While it is nice that there will be a tutorial on the program, the assessor themselves can demonstrate how the program is used before the students have to use them for an assessment. Hence, this is not a priority on our list.	Small 2 s.p.	Producing a full, easy-to-read user guide for the application can be time-consuming, therefore we are assigning 2 s.p. to it.	Upon performing a certain action, the user should be presented with an interactive guide on how to use the different components of the application.

		US-3.4	Gloria Baldwin (student)	be reminded that I'm about to lose my work	I do not lose my work when I leave my question.	Small 2 s.p.	T-3.4.1	Ask the user if they want to lose their progress if they have unsubmitted progress and try to move to another question.	Should have	If a student moves onto another question without submitting and loses their progress, it can be quite frustrating. Hence, giving the student a reminder before allowing them to move on can prevent this from happening.	Small 1 s.p.	Adding a pop-up (or similar) when the student has unsubmitted progress should be relatively straightforward.	1. When the user decides to move onto another question without submitting the current response that is non-empty, a pop-up should be displayed letting the user know they are about to lose their progress. 2. If the user decides to cancel the action, the attempt to change questions should be forfeited.
							T-3.4.2	Ask the user if they are certain they want to exit when they have unsubmitted progress and exit the page.	Will NOT have	Given all the other subtasks that have been assigned so far, we believe that this feature may be too many to work on at the moment and hence have decided for now that we will not complete this subtask. In addition, we do not believe that this brings worthwhile value to project as the user very likely wants to exit the program if they choose to exit, knowing that they will lose their progress.	Small 1 s.p.	Binding a reminder before the website is closed should be simple.	Upon the user trying to exit the page, a pop-up should be shown to confirm whether the user wants to abandon their current progress.