checkin takeoff

CITO Skills Checker: Technical Documentation

The National Adult Literacy Agency in Ireland

October 2019
Dissemination Level: Public

Co-funded by the Erasmus+ Programme of the European Union

Table of Contents

Requirements	3
Key Concepts	4
User Experience Overview	4
Solution Architecture	15
Application Stack.	21
RESTful API	25
Database	33
Component Instructions	35
Build Instructions	42
Content	44
Open Source	49
Licenses	60

Requirements

The original requirements for the CITO Skills Checker application were specified in the project tender request document. As with any project of this nature, these requirements primarily served as an overarching scope for the project, identifying the key objectives and goals of the project. More specific details of the required functionality and the user experience provided by the app were iteratively developed through a collaborative design process involving both NALA and the Learnovate Centre.

For reference, some of the key requirements from the request for tender document are included below:

The online assessment tool will be used by members of the public to assess their literacy, numeracy and digital skills and will allow them to look at their readiness to learn and learning goals. The online assessment tool will orientate them, in a clear and simple manner, to their options regarding a flexible learning opportunity as well as a pathway to recognition of their prior basic skills.

The new assessment tool aims to:

a. Improve the user experience by employing an adaptive response process that enables a user to explore their needs, recognise their existing skills and orientate them towards relevant learning options.

b. Incorporate best practice pedagogical principles, good adult literacy practice principles, universal design and plain English, with content mapped to the National Framework of Qualification (NFQ) and European Qualifications Framework (EQF) standards and levels for literacy, numeracy and digital skills.

c. Reach a wider audience, making the assessment of skills and skills needs opportunities more widely available to people with low educational attainment in Europe.

d. Increase awareness of learning options available to users in their own country and orientate the user towards these options. The assessment

tool will also show the benefits of lifelong learning in relation to personal and career development.

e. Create a valid and reliable instrument which is culturally localised for use in Ireland, Malta and Norway. The assessment tool must also be designed to be adapted to different European cultures and languages.

The new online assessment tool must be open source, mobile enabled and available as a plugin for stakeholders to use.

Our goal is to provide an easy-to-use online assessment tool which is accessible across multiple devices (desktop, mobile, tablet). The tool will allow a user to have a successful and positive experience, exploring learning goals, recognising existing skills and skills to be developed. The user will be shown relevant learning options available to them in their own country (Ireland, Malta and Norway). The tool will use real-world scenario based assessments.

Key concepts

The following concepts/terms are used throughout this document as they relate to the basic features of the CITO Skills Checker application.

Goal

A Goal describes a context in which it is situated (e.g. at home, family, the workplace) as well as a more specific situation that the user is interested in developing their skills (e.g. banking, helping with children's homework, etc.)

Skill

A skill is one of Literacy (Reading & Writing), Numeracy (Maths or Digital Skills (Computing))

Task

A Task is a specific contextualised situation that requires the use of multiple

skills to successfully navigate. A task provides the user with a use case that is grounded in a real world situation that is relevant to their daily life. The aim of a task is to allow the user to picture themselves in the situation and reflect on how well they could deal with the situation in terms of the specific skills needed.

Broader Dimensions

One of the key concepts within the app is the self-assessment by users oftheir ability to complete a real work task. To help scaffold the user's assessment, they are asked to consider the task through the lens of 3 different dimensions: confidence, independence and fluency.

Confidence

The ability to complete a task in a public context

Independence

The ability to complete a task without help

Fluency

The ability to complete a task quickly and with ease

Learning Pathway

A set of recommended learning experiences that are tailored to the specific needs of the user

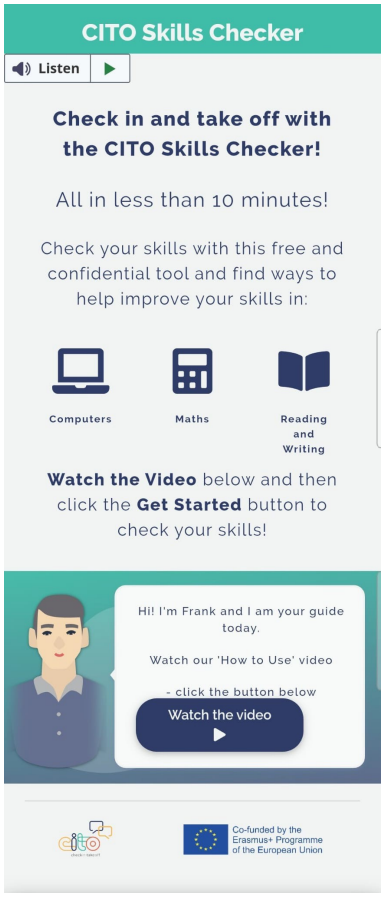
Learning Opportunity

An online or face to face course or similar learning experience that addresses one or more of the skills

User Experience

User experience overview

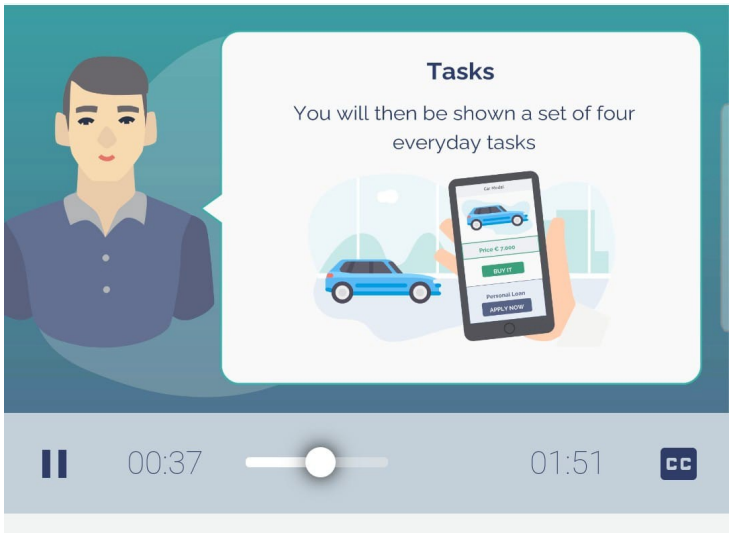
Users access the CITO Skills Checker application using the web browser on theirpreferred device (mobile, tablet, laptop or desktop).

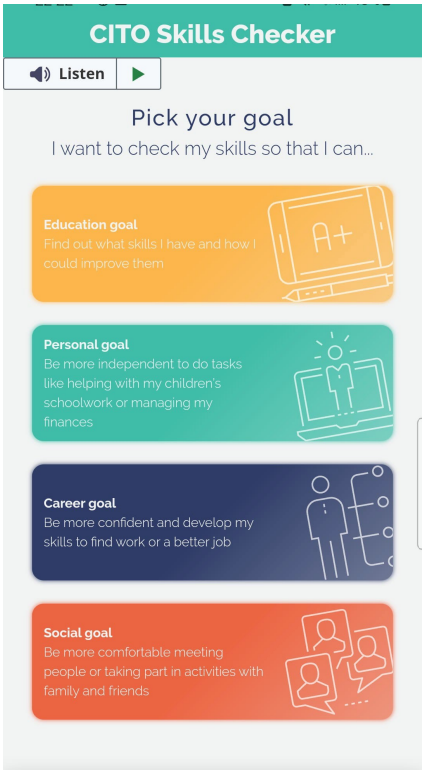


The first screen in the application is designed to provide the user with a basic understanding of what the tool is about and why the user might want to use the tool.

In the left hand corner there is a useful text-to-speech tool, ReadSpeaker, which allows the user to highlight, listen or translate selected text or the whole page.

From the home screen the user can access a short two minute 'how to' video that is designed to orient the user, showing them what they can expect from the Skills Checker experience and how they can use the tool.



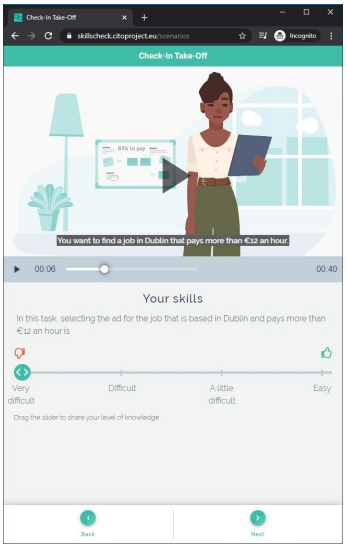


If the user chooses to start the Skills Checker experience (by clicking on the Get Started button), they are presented with a list of 4 different Goals relating to real world goals that the user might be interested in achieving. The Goals are colour coded depending on which of the 4 categories the goals fall into. The 4 categories are:

- Personal
- Economic
- Education
- Social

The Goals that are displayed on this screen can be configured independently of the application, allowing theapplication to be adapted to future needs.

On selecting a Goal, the user is presented with a set of real-world Tasks that are illustrated using animated video. The user is asked to think about how challenging they might find completing the task andwhich of the 3 skills they might find difficult.



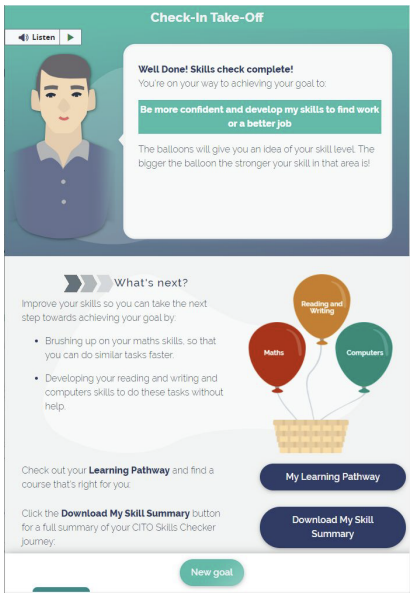
Depending on how the user answers these questions they will be asked to answer up to 7 different questions on each task.

Users will be presented with 4 different Tasks relating to the Goal thatthey selected with each task becoming increasingly more challenging. At any stage the user can choose to end the experience and go directly to the results screen, skipping any remaining Tasks.

The user is also free navigate back through the questions in a Task and toprevious Tasks that they have already considered.

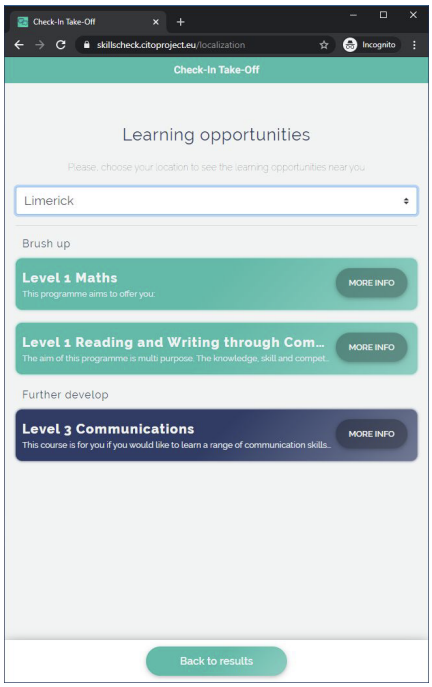
When the user has completed all 4 Tasks they are presented with the results screen. On this screen they are presented with:

- **Results visualization and explanation**



A set of 3 balloons are used to visually represent how the user self-assessed their ability to complete the tasks and which skills they found challenging. The size of the balloons relative to each other changes to illustrate this in a way that is not tied to an absolute score.

- **Learning pathway**



The Learning Pathway consists of up to 3 recommended Learning Opportunities, one for each of the 3 skills. If a user does not identify a specific skill as being challenging, then they a Learning Opportunity for that skill will not be included in the Learning Pathway. If a user is interested in a specific Learning Opportunity they can get more details about it including contact details for the providers of the Learning Opportunity.

The Learning Pathway displayed on the results screen is generated based on the individual user but at the same time it consists of Learning Opportunities that are accessible to anybody in the country irrespective of location. In other words the Learning Opportunities are generally online.

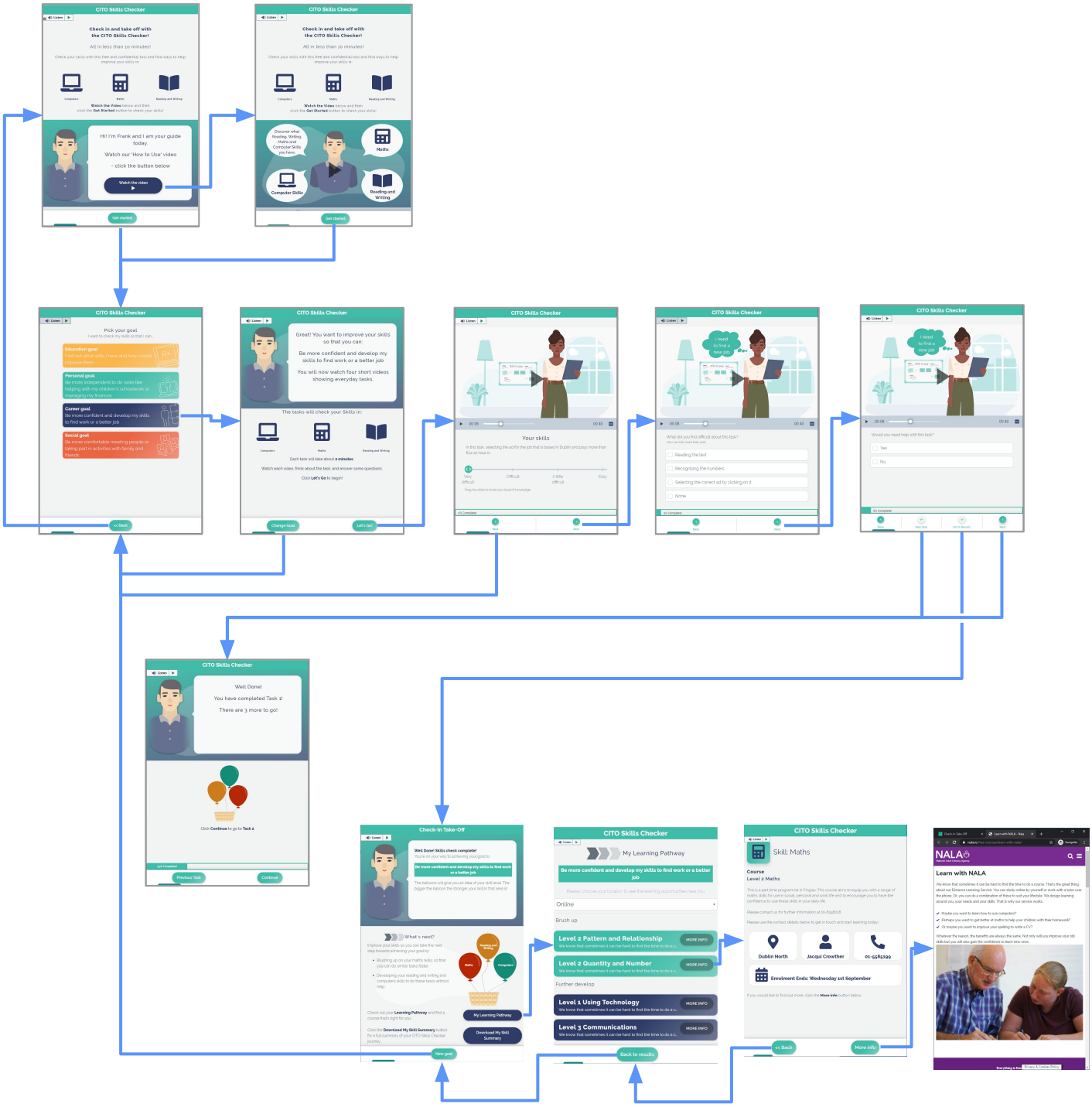
The user also has the option of generating a more customised Learning Pathway that is made up of face to face Learning Opportunities available within the users area. To achieve this, the user can select one of the predefined locations from the drop down menu. A new Learning Pathway is generated using Learning Opportunities from the user's selected area instead of more generally accessible online Learning Opportunities.

- **Skill summary download**

A Skills summary in jpg format is available for the user to download and to share with friends or family. This will allow the user to refer back to the experience at a later stage and reflect and process their skills check.

User experience flow

Here we illustrate the user experience flow of the application. The blue arrows show the navigation path between screens based on user interactions with the specific controls in the user interface.

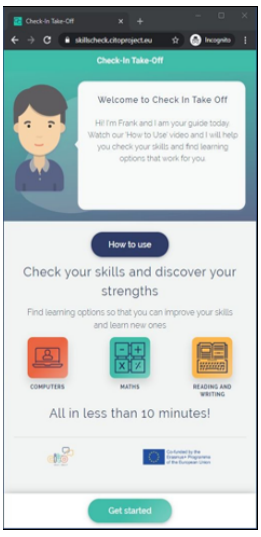


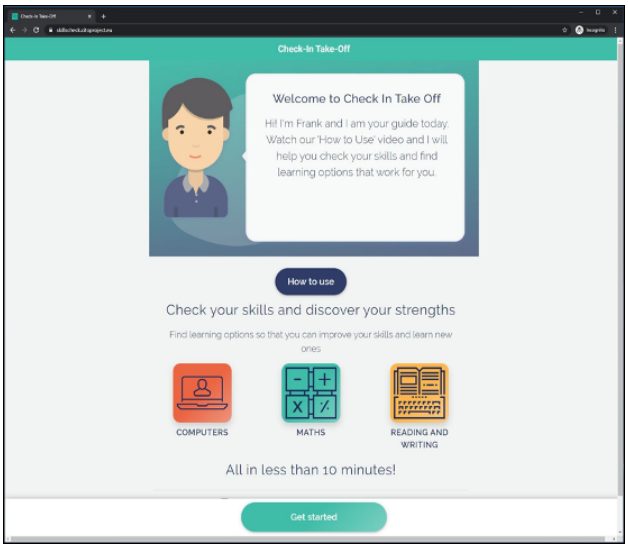
Responsive user interface design

To support the use of the Skills Checker application across a range of different devices from mobile phones and tablets to laptop/desktop devices a responsive design approach was taken in developing the user interface. This allows the UI to scale across different device form factors. In doing so, specific attention was paid to the mobile and tablet form factors while the desktop (wide screen) form factor was considered less important. The motivation for this was two fold:

- the primary device of the target user group was considered to be the a mobile phone
- tablet devices (iPads) were selected for use as part of moderated trials of the application as part of the project evaluation methodology.

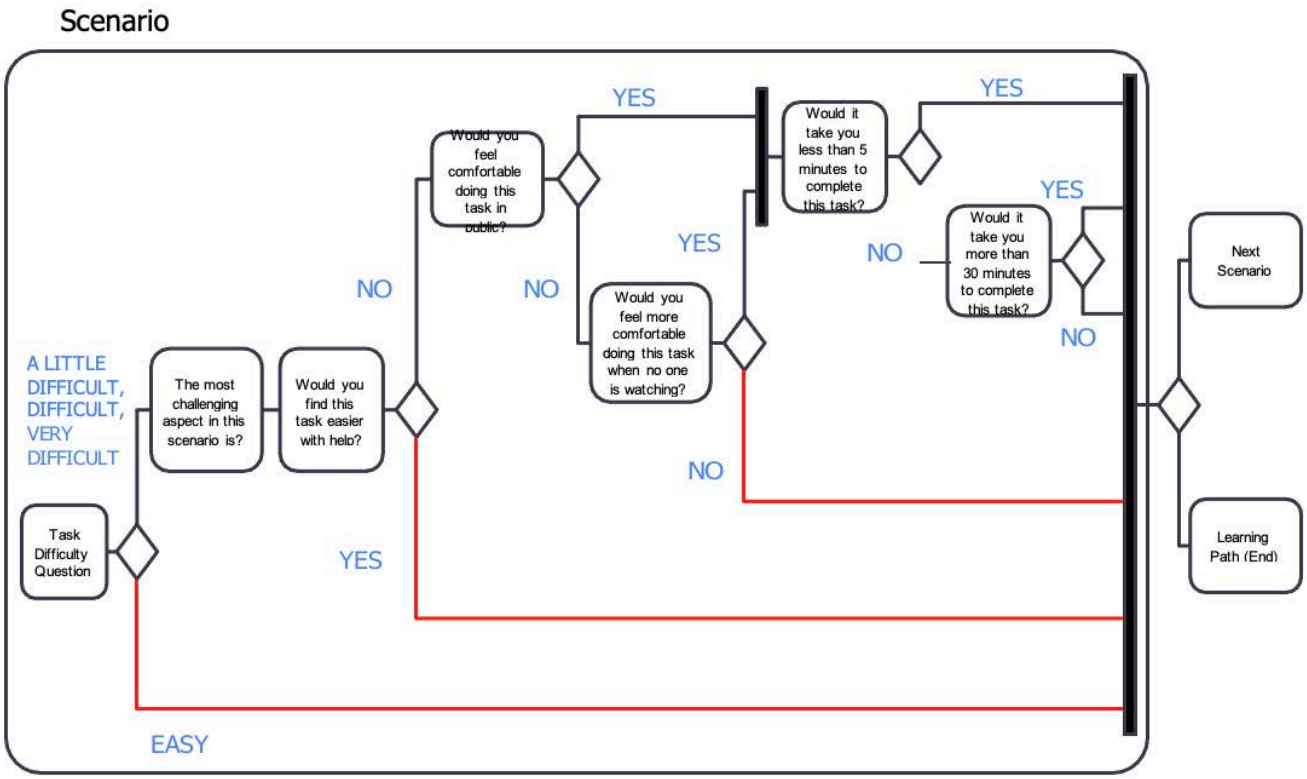
Below are two screenshots of the Skills Checker home screen for the first prototype to illustrate how the application scales depending on the form factor of the device. The screenshot on the left illustrates the layout on a typical mobile phone form factor while the screenshot on the right illustrates how the application scales to make better use of a wider form factor such as a desktop or laptop screen.





Client application - Question logic

The activity diagram below illustrates the logic that governs which questions are displayed to a user for each task. As illustrated, a user will always be presented with the first three questions. Whether or not a user will be asked to answer subsequent questions is determined by how the user answers the prior question. These conditional questions relate to the broader dimensions (Independence, Confidence and Fluency) and as such if a user answers negatively to the question then the remaining questions are not posed.

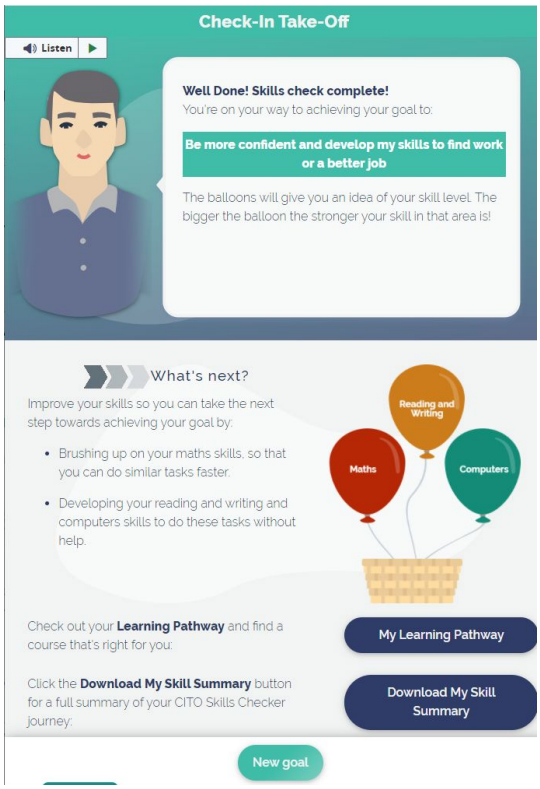


Results - Text

The results screen provides the user with both a visual representation of their skills as indicated by how they answered the questions on the tasks and a textual description.

The text component of the results consists of 3 parts:

- ·Goal
- Brush up
- Further develop



The goal statement, shown to the right of the avatar is intended to contextualise the results based on the goal that the user originally selected.

The statement is dynamically generated by combining a static sentence example:

Well done! Skills check complete! You're on your way to achieveing your goal to <goal>

To ensure that this makes sense in this context the goal is adapted from the first person (my) to the third person (your).

In addition some key words and phrases are modified dynamically to ensure that it is grammatically correct.

The user is also provided with information relating to the skills and broader dimensions that they indicated as challenging. Again these statements are dynamically generated by populating a template with the skills and broader dimensions.

Take the next step by brushing up on your <skill> and <skill> skills so that you can do similar tasks <broader dimension>
You can also reach your goal by developing your <skill> and <skill> skills to do these tasks <broader dimension>
The application can adapt these statements based on the number of skills and broader dimensions indicated by the users experience so that a single statement can mention 1, 2 or 3 skills and similarly anywhere between 0 and 3 broader dimensions.

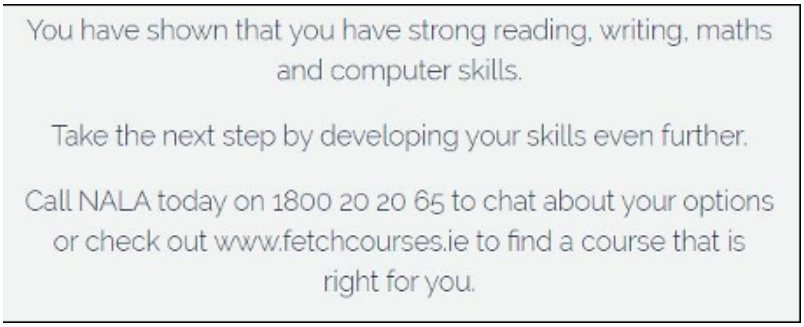
If a user happens to take a path through the task questions that results in no broader dimensions being specified then the statement will not include that portion of the statement. For example the simplest possible set of statements might look like this:

Take the next step by brushing up on your maths skills
You can also reach your goal by developing your reading and writing skills

If no skills are indicated by the user as challenging for a specific level of difficulty then the corresponding statement will not be displayed. For example, if a user selected difficult for all 4 tasks then they might see the following text (which does not include the first statement):
You can also reach your goal by developing your maths and reading and writing skills

Special Case

There is also a special case where the user indicates that they found all 4 tasks easy. In this case a special message is displayed in place of the two statements discussed here.



Results - Learning pathway

The Learning Pathway provides the user with recommended Learning Opportunities (online or face to face training courses) based on the skills that they identified as challenging. For each of the 3 skills identified by the user, the Learning Pathway will provide a single recommended Learning Opportunity. Recommended Learning Opportunities appear in one of the 2 sections of the Learning Pathway:

- Brush up
- Further develop

Brush up indicates that the user identified a skill as being challenging in relation to a task that they considered 'a little difficult'. The intended meaning here is that although the user identified the skill they have some level of ability in this skill and so only need to build upon their existing abilities with respect to this skill.

Further develop indicates that the user identified a skill as being challenging in relation to a task that they considered either 'difficult' or 'very difficult'. The intended meaning is that the user has limited or no prior ability with respect to this skill and needs to develop their skills. The recommended Learning Opportunities presented to the user each have an associated level of difficulty based on the EFQ.

The application selects a Learning Opportunity based on the level of difficulty for the task that the user first indicated that they found the skill challenging. For example, if a user indicates that they found maths challenging for a Level 2 task and then also indicated that they found maths challenging as part of the Level 3 task then they will be recommended a Learning Opportunity designed to provide a EFQ Level 2 outcome. If a user indicates that they found a task 'easy' then they will not be recommended a Learning Opportunity based on that particular task.

Solution Architecture

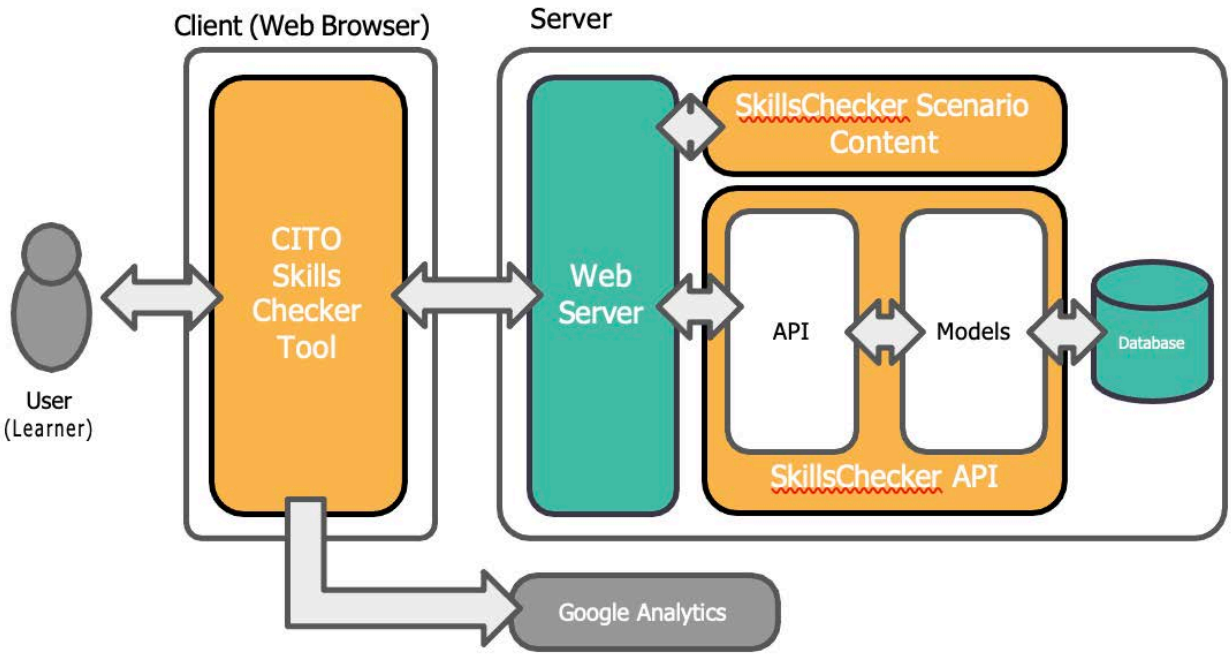
Solution Architecture

The Skills Checker application was implemented using a Client/Server architecture. The Client is responsible for the user experience and is implemented using the Angular javascript framework. This means that the application itself runs in the users web browser. One of the advantages of this approach is that it minimises the computing requirements needed to host the application. At the same time it also minimises the need for user data to be sent to and stored on the server.

The Server component provides a set of RESTful APIs (web services) that allow the Client component to dynamically retrieve information that might change over time such as the Goals, Tasks, Questions, Learning Opportunities, etc. This approach was taken with the aim of providing as much flexibility as possible for the future reuse of the Skills Checker application with a minimal level of technical effort. The Skills Checker server side component primarily acts as an interface to a relational database that is used to store all of the stateful information required by the application (goals, tasks, etc.).

The high level architecture diagram below illustrates the client/server architecture while showing some of the additional infrastructure required to support the application. The components shown in yellow represent the newly developed components and resources that make up the Skills Checker application while the components shown in green are applications such as web servers and database that provide other basic functionality.

The Skills Checker API is exposed through a web server that sits in front and exposes the API, acting as a basic API gateway. This configuration means that additional resources such as the Skills Checker tool itself and the custom developed Skills Checker content can also be served by the web server. The web server can also be configured to provide a single point at which security can be configured (SSL).



SkillsChecker Client App

Provides the whole user experience for the tool. Built as a client side (browser based) application using the Angular javascript platform.

The Skills Checker tool, running in the web browser on the user's personal device (e.g. mobile phone), will interact the a web based (RESTful) API to dynamically retrieve information about available interests, scenarios, skills, etc. This API will be hosted on the same server from which the user accessed the client application (e.g. https://skillscheck.er.ie). The main motivation for this is to ensure that the SkillsChecker tool is highly configurable and can be updated at any time to include new/different interests, scenarios, etc. without the need to modify the source code for the tool. Similarly, content used to deliver specific scenarios will be hosted on the server and loaded as needed rather than being incorporated directly into the tool itself.

Skills Checker API

Provides the client app with dynamic access to information about interests, scenarios, skills and learning opportunities that are available. This will allow the Skills Checker tool to be updated with new interests, scenarios, etc. without the need to modify the code for the tool itself.

The API will be implemented using a server side javascript framework such as Express.js and will provide a RESTful API to provide access to necessary information. The Skills Checker API will retrieve information from a relational database via a model based approach that will decouple the Skills Checker API component from the actual database technology being used. This will provide a degree of flexibility in terms of the specific database provider (PostgreSQL, Microsoft, Oracle, etc.) used.

The current scope for the Skills Checker tool does not include an administration interface so this API will only serve to provide access to information already stored in the database. It will not support adding new interests, scenarios, skills, etc. The server side component could be extended at a future date to incorporate this functionality.

Skills Checker Database

The database will store information on interests, scenarios, skills and learning opportunities that the Skills Checker tool can deliver. This information will be accessed by the Skills Checker tool (client app) via the Skills Checker API. For more information on what information will be stored in the database see the Information Model section of this document.

Restful API

The server side component of the architecture is based on the Loopback server side javascript framework. The primary function of this component is to provide dynamic access to the underlying Skills Checker data model through a RESTful API. This approach was adopted in order to future proof the Skills Checker tool, allowing changes to be made to the goals, skills, tasks, learning opportunities, etc. without the need to touch the code for either the the client application or the server side component.

Loopback is an open source framework for the development of APIs and microservices. With this in mind it was identified as the preferred technology to build the Skills Checker server side component.The Skills Check API component makes use of two key features of Loopback, its support for the creation of RESTful APIs and the use of an Object Relational Model (ORM) to describe the underlying data models and manage the storage of those data models to a relational database. The use of an ORM means that all interactions with the database are managed by the framework rather than through the use of custom SQL statements.

The Skills Checker API consists of 12 API endpoints, all of which are implemented using HTTP GET methods. The use case for the Skills Checker application does not require any user data to be stored and as such there was no need for any methods that would change the state of the underlying data.The endpoints implemented are primarily designed to drive the user experience of the client application and as such they provide access to basic information such as goals, tasks, etc.

In designing the API, one of the key considerations was the need to support different ‘instances’ or ‘products’ based on the Skills Checker application. The initial use case for this was the need to support 3 different localised versions of the Skills Checker application that would be rolled out in the 3 trial partner countries Ireland, Malta and Norway.

Technologies

Client

The client application is built on the Angular javascript framework, a modern, browser based solution for the development of web applications.

Server

The server side component is built using the Loopback javascript framework for the development of APIs and microservices.

Database

The PostgreSQL relational database is used to provide data storage. This can either be a standalone database server or a managed instance such as those provided by AWS Lightsail.

Web Server

Nginx was chosen as the preferred web server application during the development of the Skills Checker application as it is widely used and supported by many operating systems. However, any modern web server (Apache, IIS, etc.) could be used in its place as there are no specific dependencies on Nginx.

Application Stack

Solution Stack

The ‘Solution Stack’ for the Skills Checker application is illustrated below. Each column in the diagram illustrates a component of the Skills Checker application and the key technology that it is built upon:

- Green components illustrate the underlying software stack that the SkillsCheck application is built on. Generally these are 3rd party / existing software components that provide the basic functionality required such as data storage, API frameworks, etc.
- Orange indicates the components of the SkillsChecker tool itself (ie. the software that will be developed by Learnovate)

Skills Checker Client App

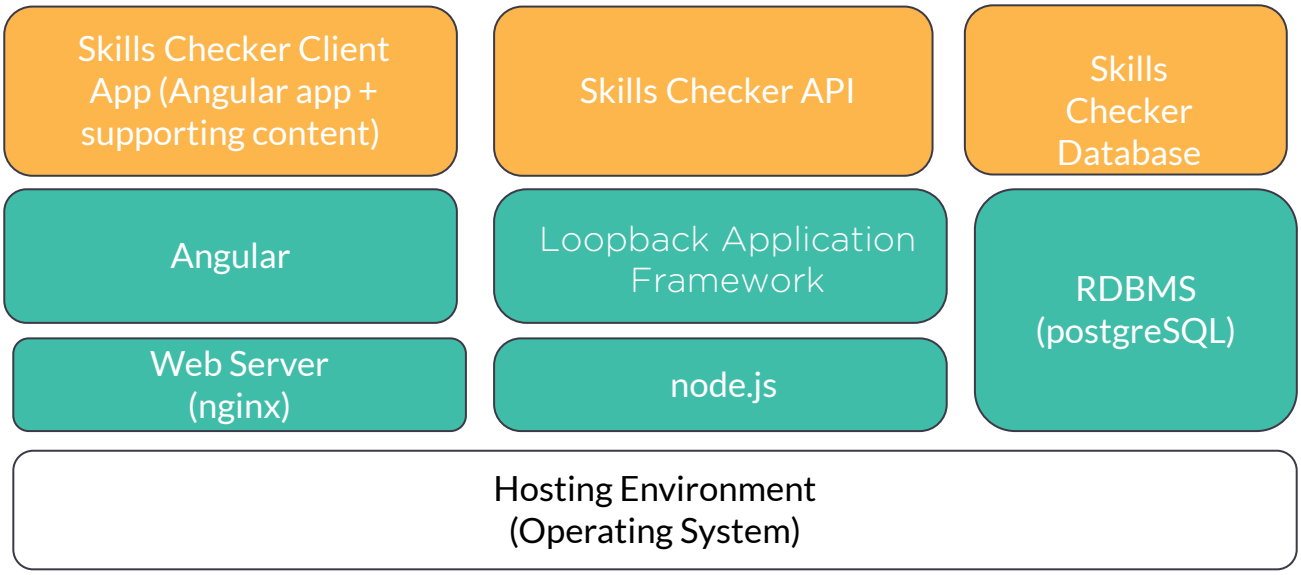
Provides the complete user experience for the application. Built as a client side (browser based) application using the Angular javascript framework. The app is made available to users online through a web server, which is used to host the application files and serve them to the client web browser where the application actually runs.

Skills Checker API

Provides the client application with dynamic access to information about goals, tasks, skills and learning opportunities that are available. This will allow the Skills Checker application to be updated with new goals, tasks, etc. without the need to modify the code for the application itself.

Skills Checker Database

Stores information on goals, tasks, skills and learning opportunities that the Skills Checker application can deliver.



Core Technologies

The 3rd party software identified are all freely available, open source technologies:

Technology	Version	Description	License
Angular	8.1.3	Javascript based platform for develop web and mobile applications	MIT License
Loopback	4.0	LoopBack makes it easy to build modern applications that require complex integrations	MIT License
Node.js	10.21.0	Server side javascript framework	MIT License
Nginx	1.14	Web Server*	2-clause BSD like License
PostgreSQL		Relational Database Server*	PostgreSQL License
Ubuntu Linux	18.04.4 LTS	AWS Virtual Machine Operating System*	-

Angular dependencies

The following table lists additional libraries that Angular depends on:

Technology	Version	Description	License
tslib	1.9.0	This is a runtime library for TypeScript that contains all of the TypeScript helper functions.	0BSD
types/chart.js	2.9.9	This package contains type definitions for Chart.js	MIT License
types/core-js	2.5.2	This package contains type definitions for core-js	MIT License
rxjs	6.4.0	RxJS is a library for composing asynchronous and event-based programs by using observable sequences.	Apache 2.0 License
normalize.css	8.0.1	Normalize.css makes browsers render all elements more consistently and in line with modern standards.	MIT License
zone.js	0.9.1	Implements Zones for JavaScript, inspired by Dart. A Zone is an execution context that persists across async tasks.	MIT License

* These software components can be readily switched for other open source or commercially available alternatives as necessary.

3rd party libraries

The following table lists additional 3rd party libraries that were used to provide functionality necessary to build the SkillsCheck client application.

Technology	Version	Description	License
angular-bootstrap-md	8.8.1	Material Design for Bootstrap (Angular version). Angular Bootstrap UI KIT	MIT License
- fortawesome/fontawesome-free	5.12.0	Official Angular component for Font Awesome 5	MIT License
- animate.css	3.7.2	Animate.css is a library of ready-to-use, cross-browser animations for use in your web projects	MIT License
- chart.js	2.9.3	Simple yet flexible JavaScript charting for designers & developers	MIT License
- hammer.js	2.0.8	Hammer is a open-source library that can recognize gestures made by touch, mouse and pointerEvents	MIT License
videogular2	7.0.1	Videogular is an HTML5 video player for Angular 2.0	MIT License
file-saver	2.0.2	FileSaver.js is the solution to saving files on the client-side	MIT License
ng5-slider	1.2.4	Self-contained, mobile-friendly slider component for Angular 5+ based on angularjs-slider	MIT License
normalize.css	8.0.1	Normalize.css makes browsers render all elements more consistently and in line with modern standards.	MIT License

RESTful API

RESTful API

A summary list of the the Skills Checker API endpoints can be found below. Those highlighted in red provide functionality that is not currently utilised by the client application but which was developed to support functionality that was subsequently removed for the application. In describing the URLs for each API endpoint we have ignored the protocol and domain name that would normally prefix each URL. Additionally we have used the convention of representing path variables using curly brackets, for example {productname} indicates that a valid unique product identifier should be used to create a valid path.

/{productname}/product
/{productname}/categories
/{productname}/categories/{categoryid}/interests
/{productname}/categories/{categoryid}/interests/{interestid}/scenarios
/{productname}/categories/{categoryid}/interests/{interestid}/scenarios/{scenarioid}/questions
/{productname}/categories/{categoryid}/interests/{interestid}/scenarios/{scenarioid}/questions/{questionid}/answers /{productname}/interests
/{productname}/interests/{interestid}/scenarios
/{productname}/interests/{interestid}/scenarios/{scenarioid}/questions
/{productname}/questionorder
/{productname}/courses
/{productname}/courses/{id}

A more detailed description of each API call, its parameterisation and expected response can be found in the tables on the subsequent pages.

GET	**/{productname}/product**	
Description:	**Parameters:**	**Example Response:**
Check that the user is accessing a valid product version	• **productname:** Name of the version that is loaded	{ "id": 1, "name": "nala", "description": "description" }

GET	**/{productname}/questionorder**	
Description:	**Parameters:**	**Example Response:**
Retrieve the order in which the questions will be asked in the scenarios	• **productname**: Name of the version from which order is being loaded	[{ "id": 1, "product": 1, "name": "task_question", "order": 1, "description": null }, ...]

GET	**/{productname}/categories**	
Description:	**Parameters:**	**Example Response:**
Retrieve the categories for a specific version	• **productname:** Name of the version from which categories are being loaded	[{ "id": 1, "name": "personal", "text": "Personal", "colour": "green", "resource": "person-laptop.svg", "description": null, "product": 1 }, ...]

GET	**/{productname}/interests**	
Description:	**Parameters:**	**Example Response:**
Retrieve the interests (goals)for a specific version	• **productname**: Name of the version from which interests are being loaded	[{ "id": 1, "name": "help_children", "text": "Help with my children's learning", "resource": "task-list-checked.svg", "description": null, "product": 1, "category": 1 }, ...]

GET	**/{productname}/categories/{categoryid}/interests**	
Description:	**Parameters:**	**Example Response:**
Retrieve the interests (goals) for a specific version and filtering by category	• **productname**: Name of the version from which interests are being loaded • **categoryid:** ID of the category that should be filter by	[{ "id": 1, "name": "help_children", "text": "Help with my children's learning", "resource": "task-list-checked.svg", "description": null, "product": 1, "category": 1 }, ...]
GET	**/{productname}/interests/{interestid}/scenarios**	
Description:	**Parameters:**	**Example Response:**
Retrieve the scenarios that will be shown to the user depending on which interest he choses	• **productname**: Name of the version from which scenarios are being loaded • **interestid**: ID of the interest that the user selected	[{ "id": 1, "name": "school_trip", "text": "School trip", "level": 1, "resource": "scenario-01.mp4", "description": "Description", "product": 1, "interest": 1 }, ...]
GET	**/{productname}/categories/{categoryid}/interests/{interestid}/scenarios**	
Description:	**Parameters:**	**Example Response:**
Retrieve the scenarios that will be shown to the user depending on which interest he choses. Also requiring category	• **productname**: Name of the version from which the scenarios are being loaded • **categoryid**: ID of the category that should be filter by • **interestid**: ID of the interest that the user selected	[{ "id": 1, "name": "school_trip", "text": "School trip", "level": 1, "resource": "scenario-01.mp4", "description": "Description", "product": 1, "interest": 1 }, ...]

GET	**/{productname}/interests/{interestid}/scenarios/{scenarioid}/questions**	
Description:	**Parameters:**	**Example Response:**
Retrieve the questions (with answers included) for a specific scenario	• **productname**: Name of the version from which the questions are being loaded • **interestid**: ID of the interest that the user selected • **scenarioid**: ID of the scenario that the user will answer next	[{ "id": 1, "type": "slider", "pedagogical_type": "task_question", "question": "In this task, selecting appropriate clothes for my child after looking at the weather forecast on my phone is", "description": null, "product": 1, "scenario": 1, "answers": [{ "id": 1, "text": "Very difficult", "value": 0, "order": 0, "special": null, "skipTo": null, "product": 1, "question": 1 }, ...] }, ...]

GET	**/{productname}/categories/{categoryid}/interests/{interestid}/scenarios/{scenarioid}/questions**	
Description:	**Parameters:**	**Example Response:**
Retrieve the questions for a specific scenario. Also requiring category	• **productname**: Name of the version from which the questions are being loaded • **categoryid**: ID of the category that should be filter by • **interestid**: ID of the interest that the user selected • **scenarioid**: ID of the scenario that the user will answer next	[{ "id": 1, "type": "slider", "pedagogical_type": "task_question", "question": "In this task, selecting appropriate clothes for my child after looking at the weather forecast on my phone is", "description": null, "product": 1, "scenario": 1, "answers": [{ "id": 1, "text": "Very difficult", "value": 0, "order": 0, "special": null, "skipTo": null, "product": 1, "question": 1 }, ...] }, ...]

GET	**/{productname}/categories/{categoryid}/interests/{interestid}/scenarios/{scenarioid}/questions/{questionid}/answers**	
Description:	**Parameters:**	**Example Request:**
Retrieve the answers for a specific question	• **productname**: Name of the version from which the questions are being loaded • **interestid**: ID of the interest that the user selected • **scenarioid**: ID of the scenario that the user will answer next	[{ "id": 1, "text": "Very difficult", "value": 0, "order": 0, "special": null, "skipTo": null, "product": 1, "question": 1 }, ...]

GET	/{productname}/courses		
Description:		**Parameters:**	**Example Request:**
Retrieve a list of courses filtering the skill levels. It is also possible to filter by location but it is optional		• **productname**: Name of the version from which the courses are being loaded • **literacyLvl**: (1 to 4) level of literacy skill to filter by • **numeracyLvl**: (1 to 4) level of numeracy skill to filter by • **digitalSkillsLvl**: (1 to 4) level of digital skill to filter by • **location**: location to filter by	[{ "id": 1, "external_id": "1234", "title": "Title", "description": "Description", "level": 1, "skill": "numeracy", "location": "online", "address": "", "link": "link", "enrolment_start": "2020-07-02T20:10:22.106Z", "enrolment_finish": "2020-07-02T20:10:22.106Z", "contact_person": "Name", "contact_telephone": "678912354", "contact_email": "email", "contact_attention": "9:00 AM to 5:00 PM", "product": 1 }, ...]
GET	**/{productname}/courses/{id}**		
Description:		**Parameters:**	**Example Request:**

Retrieve a specific course by providing the ID		• **productname**: Name of the version from which the course is being loaded
		• **courseid**: ID of the course that wants to be retrieved
			{
			"id": 1,
			"external_id": "1234",
			"title": "Title",
			"description": "Description",
			"level": 1,
			"skill": "numeracy",
			"location": "online",
			"address": "",
			"link": "link",
			"enrolment_start": "2020-07-02T20:10:22.106Z",
			"enrolment_finish": "2020-07-02T20:10:22.106Z",
			"contact_person": "Name",
			"contact_telephone": "678912354",
			"contact_email": "email",
			"contact_attention": "9:00 AM to 5:00 PM",
			"product": 1
			}

API Component - Project Structure

The figure below illustrates the structure of the Skills Checker API project. It follows a standard Loopback project structure consisting of:

- Controllers - the implementation of the RESTful API
- Datasources - Configuration of the underlying database connection
- Models - Representation of the data models as part of the ORM
- Repositories - Provides the 'connection' between the Model and the Datasource

As shown, it is quite a simple application thanks to the basic functionality that Loopback provides. Essentially there is a model, repository and controller for each of the different data models (tables) that have been implemented.

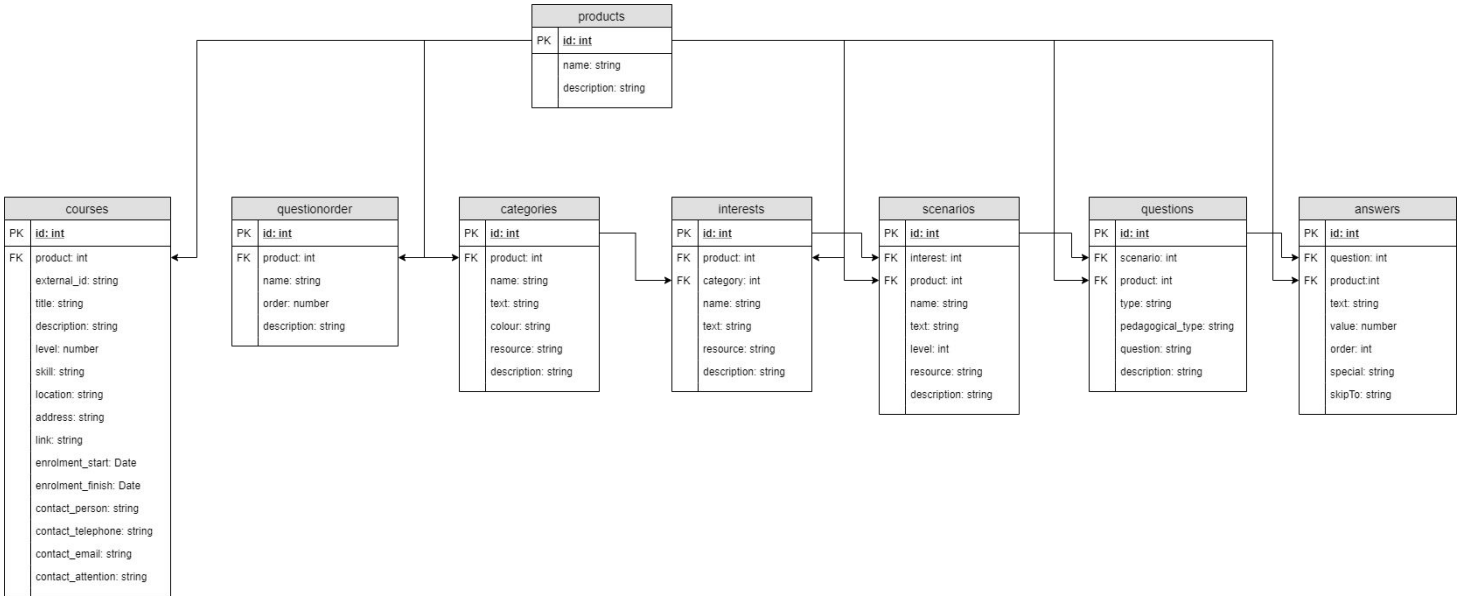
See the following section for further description of the data model.

```
- api
    - src
        - controllers
            - answer.controller.ts
            - category.controller.ts
            - common.controller.ts
            - course.controller.ts
            - interest.controller.ts
            - product.controller.ts
            - question-order.controller.ts
            - question.controller.ts
            - scenario.controller.ts
        - datasources
            - postgresql.datasource.json
            - postgresql.datasource.ts
        - models
            - answer.model.ts
            - category.model.ts
            - course.model.ts
            - interest.model.ts
            - product.model.ts
            - question-order.model.ts
            - question.model.ts
            - scenario.model.ts
        - repositories
            - answer.repository.ts
            - category.repository.ts
            - course.repository.ts
            - interest.repository.ts
            - product.repository.ts
            - question-order.repository.ts
            - question.repository.ts
            - scenario.repository.ts
```

Database

Database Models /Schema

Model / Table	Description
products	Differentiates between different 'instances' or 'versions' of SkillsCheck. For example, 3 different products were defined to represent the 3 versions of SkillsCheck for the 3 trail countries (Ireland, Malta and Norway)
categories	The is a legacy concept within the data model that is no longer used to the extent that it was intended to. Categories in the current implementation are used to categorise 'interests' (goals) into personal, economic, educational or social.
interests	Interest was the original name used for what are now called goals.
scenarios	Scenario was the original name for what are now called tasks. The Scenarios model defines the NQF level of a task, the illustrative animation/video resource, etc.
questions	The different questions that are associated with a Scenario (task). The question model specifies they type of question (slider, multiple choice, etc.) as well as the actual question text
questionorder	
answers	The answers that are associated with a specific question
courses	The Learning Opportunities used to generate the Learning Pathway. Basic information about Learning Opportunities such as titles, descriptions, etc. as well as the location of the course.



Component Interactions

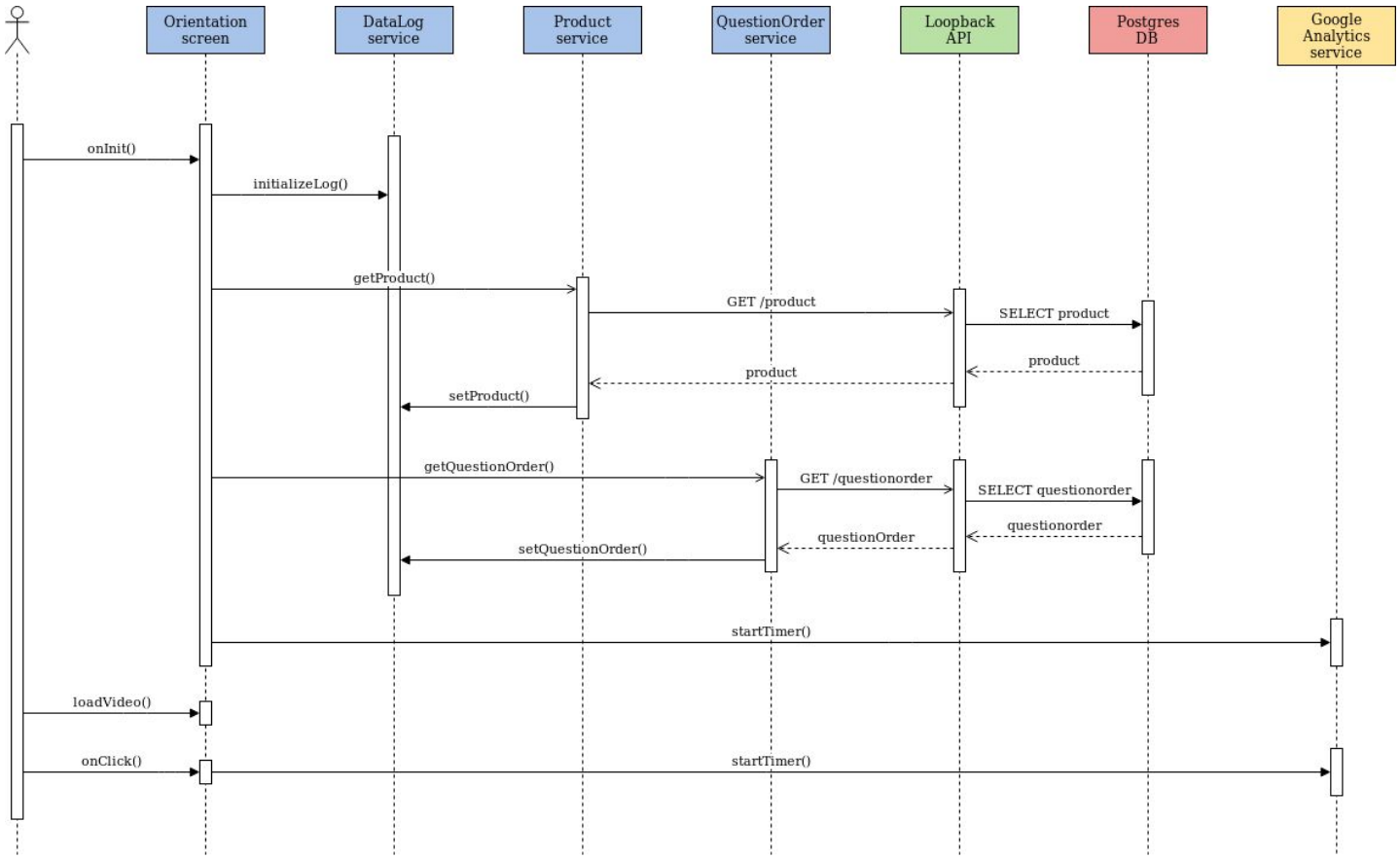
Interactions

Much of the functionality of the Skills Checker application requires interactions not only between components within the Skills Checker client application but also with the server side API and the underlying database. The interactions have been documented in the form of UML activity diagrams, as illustrated below.

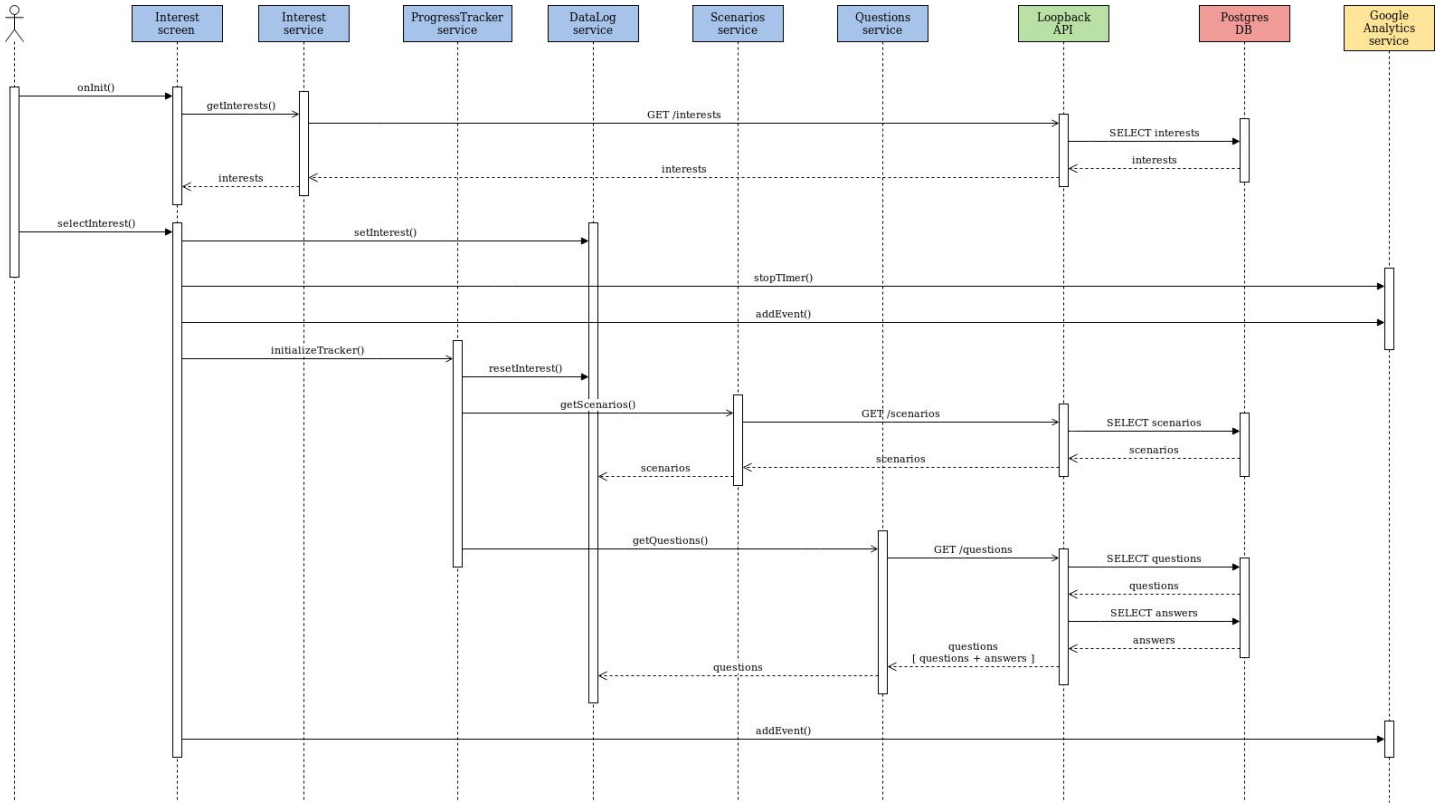
To keep these diagrams as simple as possible the interactions between the various components have been broken up into smaller logical sequences, generally representing the interactions required to generate a single page/screen within the application.

Colour coding has been used to differentiate between components within the client application (blue), the server side API (green), the database (red) and the Google Analytics service (yellow).

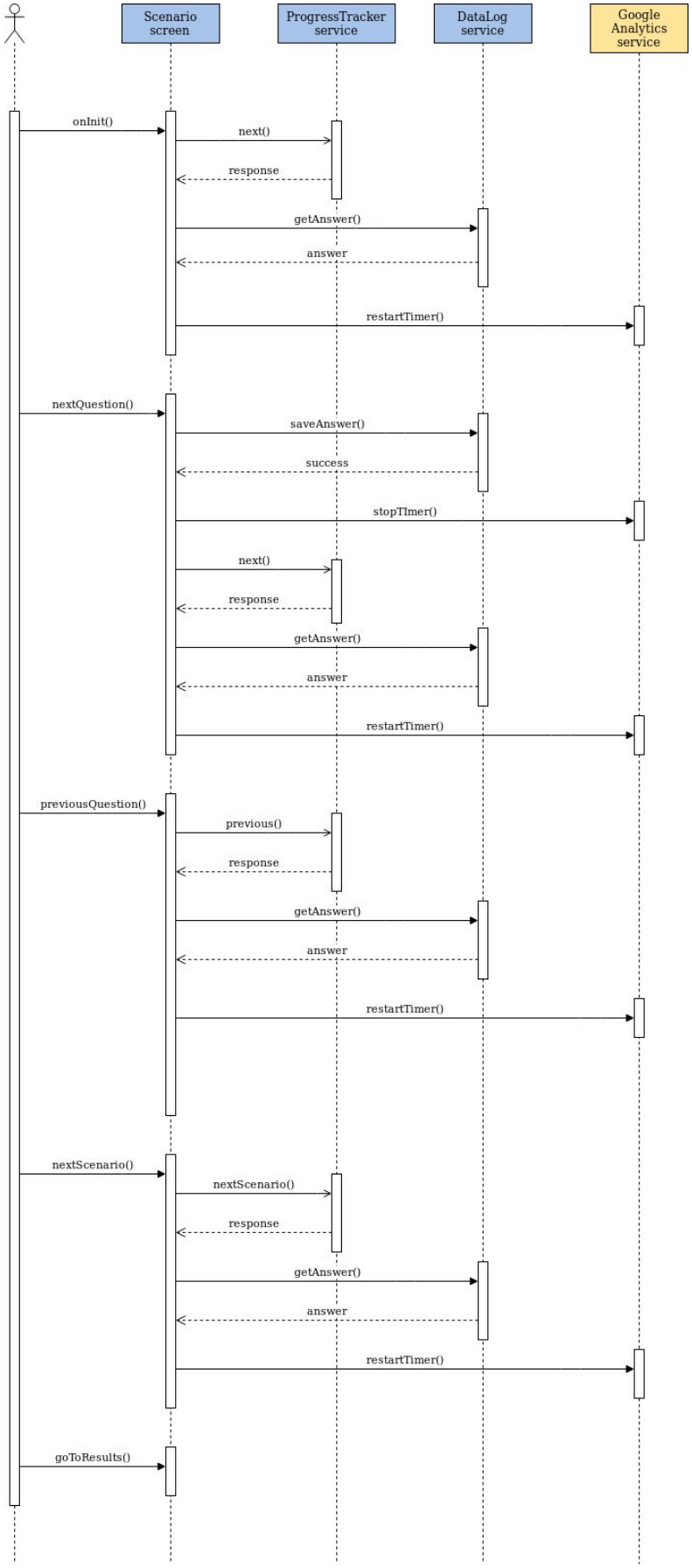
Orientation Screen



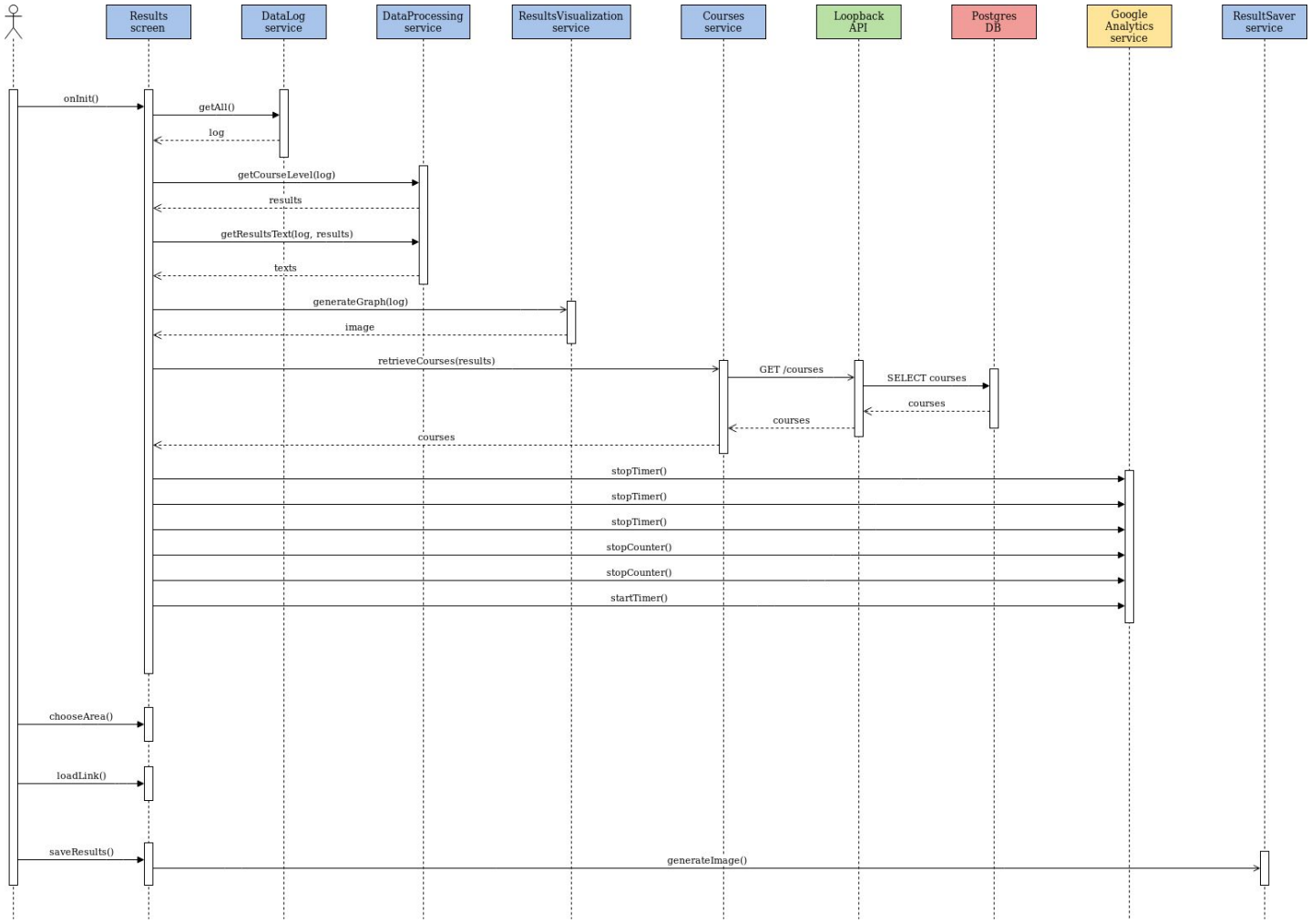
Goal (Interest) Screen



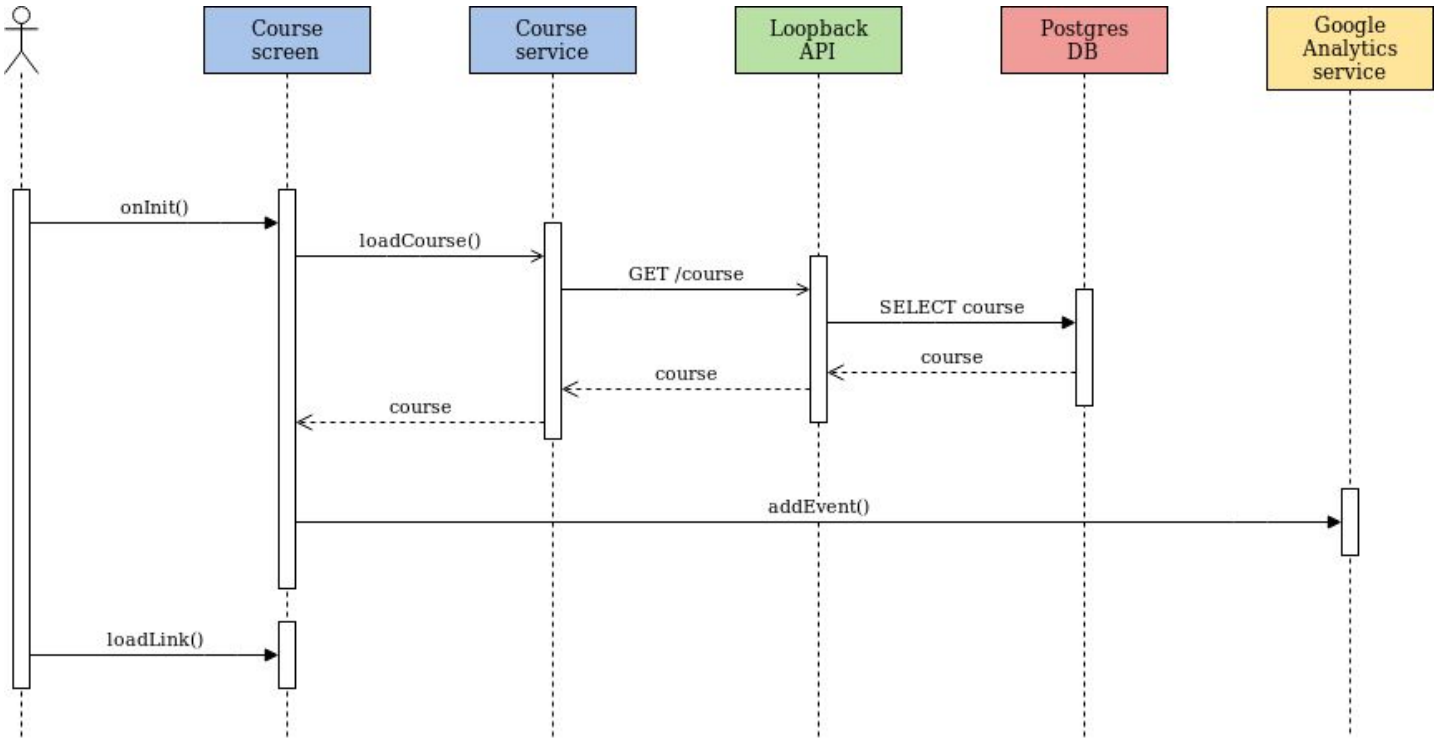
Task Screen


Scenario screen
ProgressTracker service
DataLog service
Google Analytics service
onInit()
next()
response
getAnswer()
answer
restartTimer()
nextQuestion()
saveAnswer()
success
stopTImer()
next()
response
getAnswer()
answer
restartTimer()
previousQuestion()
previous()
response
getAnswer()
answer
restartTimer()
nextScenario()
nextScenario()
response
getAnswer()
answer
restartTimer()
goToResults()

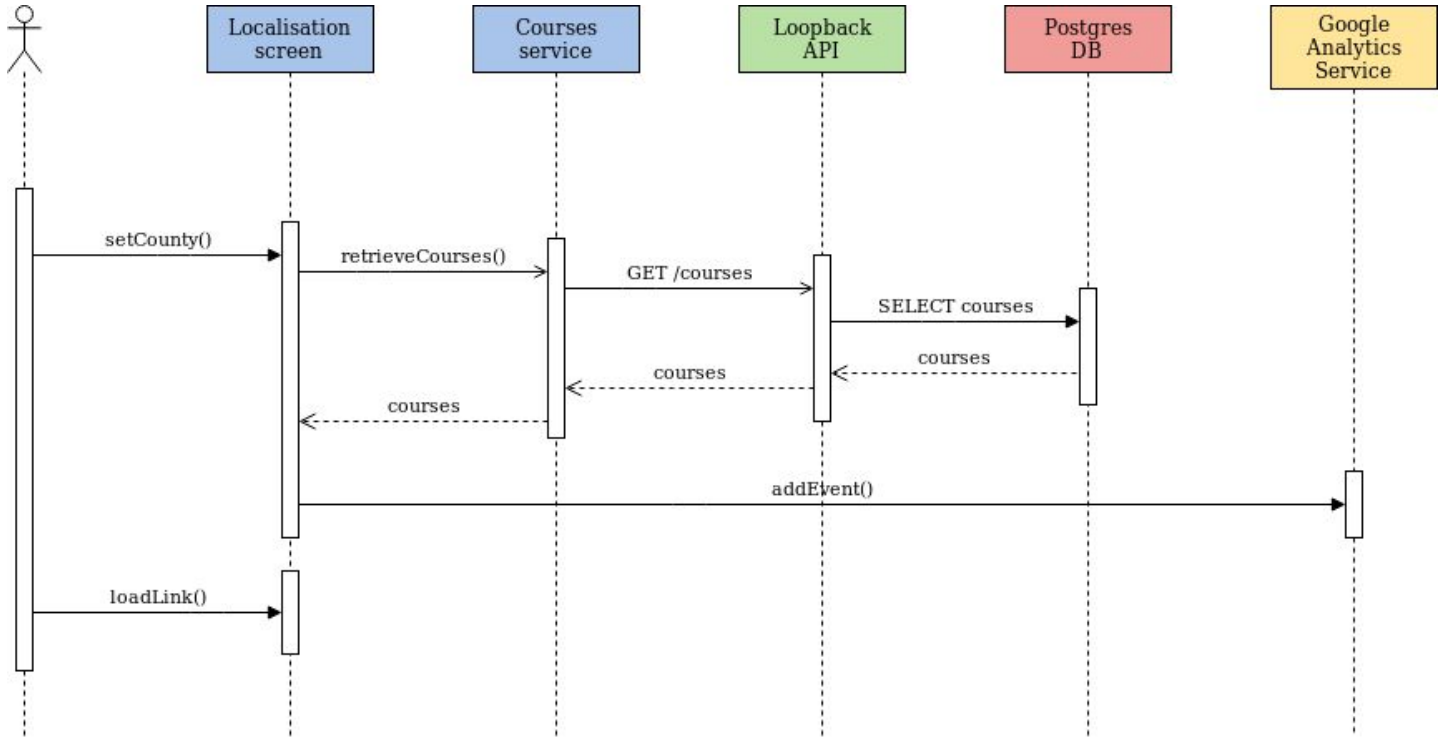
Results Screen



Courses Screen



Local Learning Pathway Screen



Client Application - Project Structure

The figure below describes the high level project structure for the Skills Checker client application (Angular). The overall structure is generally consistent with the standard layout for an Angular application with the main source files found in the src directory, which is further subdivided into app, assets, environments and style directories.

- app - contains the source files for the Angular UI components, utility services and models
- assets - contains resources such as icons, graphics and fonts used in the application
- environment - configuration files for the application
- style - configuration for the overarching look and feel used across the application

```
- src
    - app
       - components
             -   course
             -   graphics
                   - balloons-and-basket
             -   header
             -   media
             -   navigate-button
             -   question
             -   screens
                   - categories-screen
                   - course-screen
                   - how-to-screen
                   - interests-screen
                   - localization-screen
                   - orientation-screen
                   - results-screen
                   - scenario-screen
                   - scenario-introduction-screen
                   - scenarios-screen
       -   models
       -   services
             -   api-call
             -   data
             -   etc
    - assets
        - balloons
        - fonts
        - icons
        - Images
    - environments
    - style
```

Build Instructions

Build Instructions

This section describes the steps required to build and run both the client and server components of the Skills Checker application in a local environment.

Prerequisites

- Git (https://git-scm.com/)
- Node.js (https://nodejs.org/)
- NPM
- PostgreSQL (https://www.postgresql.org/)

Client Application build and installation

Retrieve code from version control (requires authentication)

```
git clone https://gitlab.scss.tcd.ie/learnovate-centre/nala-skillschecker/client-app.git
```

Install required packages

```
npm install
```

Deploy the app locally

```
npm start
```

Client will be available in `localhost:4200` by default

Build production package of application

```
ng build --prod --aot --build-optimizer --delete-output-path --optimization --base-href=/
```

Server Side (API) build and installation

Getting the code (needs authentication)

```
git clone https://gitlab.scss.tcd.ie/learnovate-centre/nala-skillschecker/api.git
```

Install required packages

```
npm install
```

Update RDBMS connection settings in `src/datasources/postgresql.datasource.json`

```
{
        "name": "postgresql",
        "connector": "postgresql",
        "url": "",
        "host": "YOUR DB HOST HERE",
        "port": YOUR PORT HERE (default: 5432),
        "user": "DB USER HERE",
        "password": "DB USER PASSWORD HERE",
        "database": "DB NAME HERE"
}
```

Build the Loopback app

```
npm build
```

If the database is not created yet or is not updated, create the schema with Loopback

```
npm run migrate --rebuild
```

Launch it

```
npm start
```

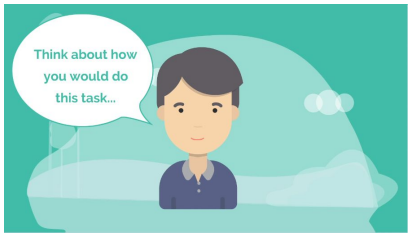
Content

Content

The Skills Checker tool is designed to help users to self assess their skills through the use of range of goal specific scenarios or ‘tasks’. Each task consists of a real world scenario that the user is asked to consider and rate how easy or difficult they would find completing that task in their daily lives.

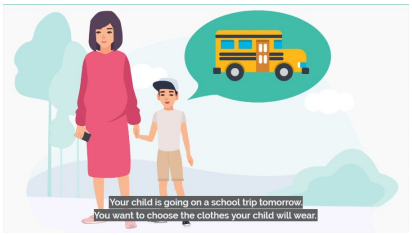
The Skills Checker application currently includes 4 different ‘goals’, across 4 different contexts (personal, economic, education and social). This means that there are a total of 16 different tasks, each of which has an associated animated scenario. Each animation is realised as an MP4 video file that includes both visual elements and a audio track that provides a voice over that describes the scenario and explains the problem that the user is being asked to consider. In addition to the video file, there are associated subtitles that are captured in separate VTT files and played back with the video.

Each task animation follows a standard pattern consisting of 4 parts:



Part 1:

Introduce the task and remind the user what they are being asked to think about.

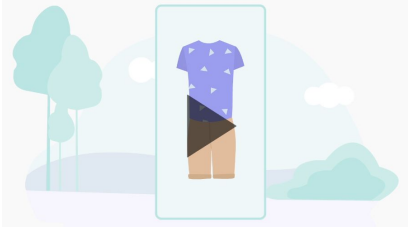


Part 2:

Task - visual and narrated description of the task that the user is being asked to consider.

Part 3:

Reflection - the video ‘pauses’ for several seconds to encourage the user to think about the task that they have just seen.



Part 4:

Outcome - illustrates how the task previously shown can be completed.

The table below lists all of the content developed for the Skills Checker application, the length of each animated video and the file name for the Irish English version of the animation.

The duration quoted for each animation may differ slightly for the other localised versions of each animation but should be broadly consistent with the times listed. The naming convention used for the file names is <category>-scenario-<number>-<locale>.mp4

Animations were authored in Adobe Animate and rendered as MP4 videos at 24fps with a data rate of 920kbps. All rendered animations have a resolution of 1494 x 840. This resolution was considered to be appropriate for use during the CITO project trials as iPads were chosen by the project partners as the device that would be used. Additionally the higher resolution provides users on larger form factor devices such as laptops or desktop with an improved experience as it removes any blurriness that might result from scaling lower resolution video files.

The impact on mobile users, who would need to download these higher resolution videos via a mobile network, was considered to be relatively low

	Category	Description	Duration	Filename (Irish English)
1	Personal	Choose your child's clothes for school trip	0:37	personal-scenario-01-en-ie.mp4
2	Personal	Invite your friends to a party	0:44	personal-scenario-02-en-ie.mp4
3	Personal	Find out how much you can borrow	0:45	personal-scenario-03-en-ie.mp4
4	Personal	Calculate the interest charged on your bill	0:47	personal-scenario-04-en-ie.mp4
5	Economic	Find a job in Dublin that pays more than €12	0:40	economic-scenario-01-en-ie.mp4
6	Economic	Create an email account to register on a jobs site	0:34	economic-scenario-02-en-ie.mp4
7	Economic	Find a bus driver job online and calculate pay	0:51	economic-scenario-03-en-ie.mp4
8	Economic	Analyse a graph and prepare a presentation	0:58	economic-scenario-04-en-ie.mp4
9	Education	Apply online for a course	0:36	education-scenario-01-en-ie.mp4
10	Education	Find an evening course in your area	0:45	education-scenario-02-en-ie.mp4
11	Education	Book and pay for your drivers theory test	1:22	education-scenario-03-en-ie.mp4
12	Education	Find the best value deal for a textbook	0:56	education-scenario-04-en-ie.mp4
13	Social	Select the bus that will allow you to arrive on time	0:48	social-scenario-01-en-ie.mp4
14	Social	Select the month with the least chance of rain	0:43	social-scenario-02-en-ie.mp4
15	Social	Adjust the ingredients for a recipe	1:05	social-scenario-03-en-ie.mp4
16	Social	Calculate how much decking you need	1:16	social-scenario-04-en-ie.mp4
17	How To	How to use the SkillsCheck application	149	how-to-use-en-ie.mp4

Subtitles

Rather than being integrated directly into the animated video files, the subtitles for each video are saved in an individual text files using the Web Video Text Tracks Format. This allows the wording and timing of the subtitles to be changed without the need to edit and render the video file itself. This approach was originally adopted with the aim of minimising the need to localise each individual video file to each country (Ireland, Norway and Malta).

The VTT files are retrieved by the Skills Checker when loading a video and played in parallel. If necessary, the styling of the subtitles can be adapted using CSS to adapt them to the specific needs of end users. The use of a separate captioning file in this way also provides additional flexibility, allowing the video to be played without the captions if/when appropriate.

The default behaviour of the application is the look for a .VTT file with the same file name as the video being retrieved. For example if the application loads personal-scenario-01-en-ie.mp4 it will attempt to retrieve the associated subtitles file personal-scenario-01-en-ie.vtt. In addition to being supported by the Skills Checker application, VTT files are also supported by many online video platforms including YouTube.

Content - Subtitles - Example VTT file

Below is an example VTT file for personal-scenario-01.
There are 3 different captions used for this video. Each caption has an associated start and end time. These are interpreted by the video player and used to determine when the caption should appear and disappear. Multiple lines of captioning can be displayed at any given time as illustrated by the first caption in the example.

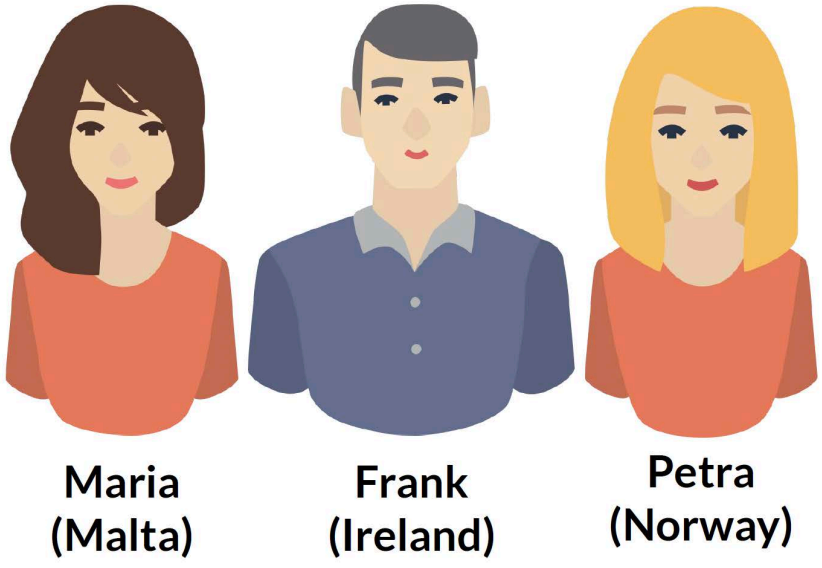
```
WEBVTT

1
00:00:06.200 --> 00:00:12.800
Your child is going on a school trip tomorrow.
You want to choose the clothes your child will wear.

2
00:00:15.300 --> 00:00:17.700
Look at tomorrow's weather forecast on your phone...

3
00:00:18.000 --> 00:00:21.400
...and decide if you can select the appropriate clothing for your child.
```

Content localisation

Each of the 17 animations were localised to the 3 countries in which the Skills Checker application will be trialed as part of the CITO project. This resulted in the creation of 51 individual animations.

The main concerns when localising the animations included:

- Language: Any text elements in the animations such as timetables, job adverts, etc. were localised both by translating into native languages (Norwegian) and also the preferred phrasing of the country in question (Malta). The narration for each video was also specific to each country with different recordings being made for Ireland, Norway and Malta. In the case of the Irish voice overs these were carried out by Learnovate where as the Norwegian and Maltese voice overs were provided by the project partners in those countries.
- Names, Locations, Currency: The names and locations used in the scenarios illustrated by the animations were also adapted to the specific countries
- Persona: The persona used to represent the application was also adapted both to match the voice over but also to align with the country in question (e.g. Maria, the persona for the Maltese versions of the animations was given dark hair)



Open Source

Stand alone web application

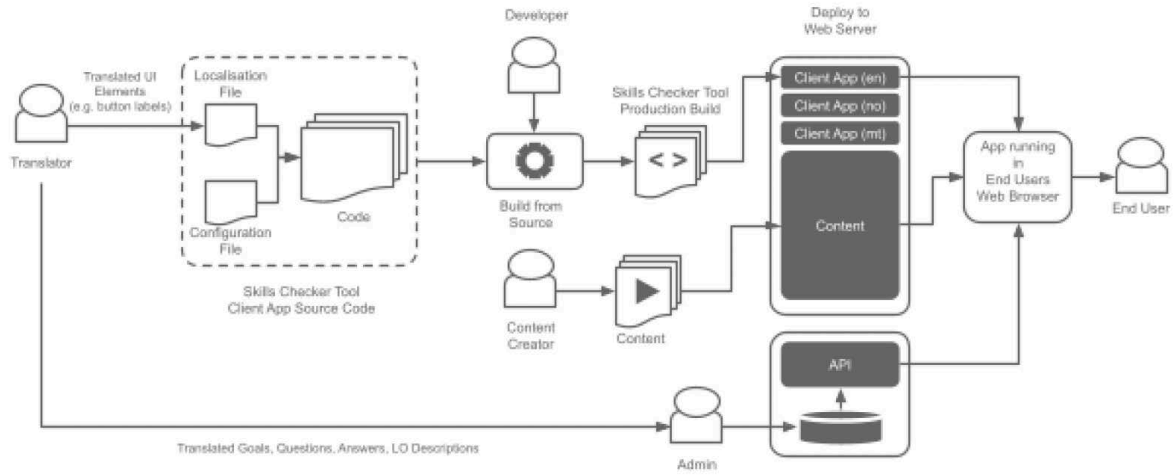
The CITO Skills Checker Tool was designed and built to be a stand alone web application that can run independently of any 3rd party environments such as the websites of training providers. This means that an end user can access the application directly providing they have the appropriate web address (e.g. https://skillschecker.ie). The application can also be embedded in a 3rd party website, for example through the use of a simple HTML iframe.

During the CITO project this embedded iframe approach was tested and used on the CITO project website. In this case the Skills Checker was embedded in a site running on a Wordpress CMS. The process was implemented by NALA, which not only allowed the ability to embed the tool into a site to be verified but also to confirm that specialist technical skills beyond a level of experience with the CMS was not necessary. More recently this was replaced with direct links to the different versions of the tool on the CITO project site. A limitation of this embedding use case is that it assumes the Skills Checker Tool is being hosted and maintained by an organisation other than the stakeholder organisation rather than the stakeholder themselves. If the stakeholder is customising and deploying their own version of the Skills Checker Tool then they will have much more control over the hosting and integration of the tool with their existing environment.

Data driven design

The CITO Skills Checker Tool was developed to be data driven. This allows for significant portions of the user experience to be customised simply through modifications to the underlying database. For instance the Goals, associated Scenarios and Questions are all defined in the database in addition to the Learning Opportunities recommended by the tool. This means that additional Goals and/or Scenarios can be added to the tool without the need to make changes to the source code of the tool itself. Other use cases for such data driven changes might be the rephrasing of existing questions and/or answers or the updating of available learning opportunities.

Throughout the course of the project 4 different instances of the CITO Skills Checker Tool were developed, one for each of the different locales being targeted by the project (Irish-English, Maltese-English, Maltese and Norwegian). To create these different instances of the tool a single translation file is modified to incorporate the required translations. This translation file is used at build time to generate a localised version of the tool. Figure 1 provides an illustration of this pipeline. This combined with a simple configuration file allows features of the tool to be enabled or disabled (e.g. the ReadSpeaker or Google Analytics) with relatively minimal effort from a software developer. This minimises the technical effort required to translate and adapt the application for different locales.

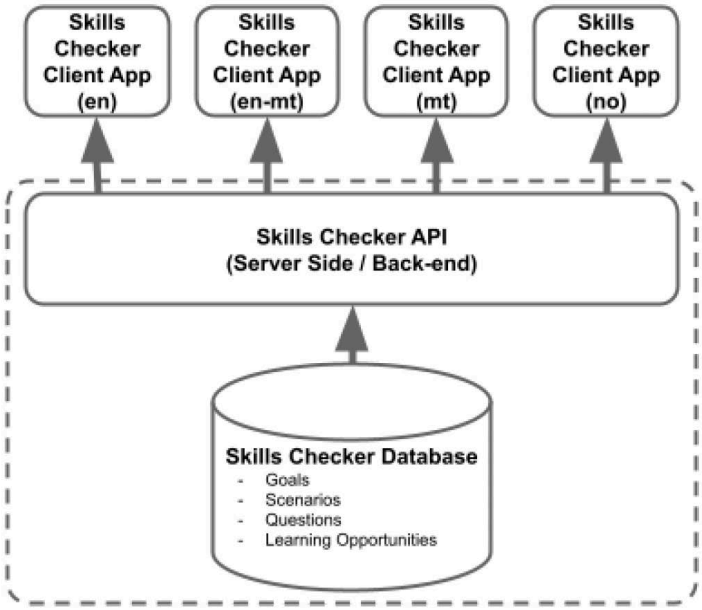


It is also worth considering that the data driven customisation can also be used to adapt the Skills Checker Tool to different contexts or environments even when using the same language. The 4 different instances of the tool all communicate with the same single back-end component (server side API) that handles access to the underlying database. The functionality provided by this back-end component was designed from the ground up to support different types of skill check tool running on top of it. As part of the CITO project, this flexibility was used to support all 4 different languages at the same time.

This is illustrated in the figure below. However, this capability could equally be used to support a range of different assessment experiences for different end users. Some possible use cases for this might include:

- A provider who wants to use the Skills Checker Tool but does not want to incorporate questions about the broader dimensions (independence, confidence and fluency). In this use case the provider could update the database to remove these specific questions. The tool would then only show the question relating to the 3 skills (reading and writing, maths and computers).
- A provider who wants their users to attempt 3 scenarios instead of 4
- A provider who wants to include additional goals or rename the existing goals. The back-end component can facilitate this type of customisation for one instance without affecting other instances running on the same platform.

All three of these example use cases could be supported and delivered by a single provider at the same time by adding the necessary information to the database and only making a very minor change to the Skills Checker Tool client app configuration to allow it to uniquely identify itself so that the correct information is served to it. If localisation of the app to support an additional language is not required then this configuration change is all that would be needed to serve a different experience for a different user group.



In the case where each provider is hosting and managing their own instance of the Skills Checker Tool then these illustrative examples are less compelling. However, if we envisage a provider who manages and maintains multiple instances of the Skills Checker Tool that need to be customised and adapted to different needs then the flexibility of the API and its support for different 'products' or 'instances' becomes more relevant.

Another advantage of the predominantly data driven approach is that it allows for real time updates or changes to be made to the tool without the need to involve software developers in rebuilding and redeploying the tool itself. Necessary changes can be made to the database and become live in the tool immediately. Use cases where this might become relevant include the maintenance of the Learning Opportunities Database (LOD), correction of spelling mistakes, addition of new Goals or Challenges, etc.

For the CITO project 16 Scenario animations were created and subsequently localised into the different languages. In the future new scenario animations can be created and added to the tool through the database. These scenario animations are not fundamentally incorporated or 'hard coded' into the tool in any way allowing them to be easily replaced with scenarios or animations that are appropriate for different contexts or to meet the changing needs of users.

As such, the CITO Skills Checker Tool can be seen as a general purpose tool for assessing skills through scenarios. Although some aspects of the tool, such as the messaging around the 3 literacy skills would need to be modified for a different use case, the core functionality of selecting a goal and answering questions about a series of tasks can support a wide range of different assessment related use cases.

The Skills Checker Tool consists of 3 components:

- Client Application that runs in the users web browser
- Server Component (REST API) that runs on the hosting environment
- Database

All 3 components are built on an open source technology stack summarised below. For additional details of the application stack and associated licenses please see the previous sections of the report:

- Client Application
- Angular - MIT License
- Server Component
- Loopback - MIT License
- Database
- PostgreSQL - PostgreSQL License

As listed above the technology stack used to develop the Skills Checker tool was selected not only for its technical capabilities but also based on the open and flexible nature of the licenses under which it is made available. Although Learnovate did not bundle the underlying application stack with the Skills Checker Tool, they felt that it was important to ensure that there was a range of 3rd party extensions and libraries available that were not encumbered by any proprietary licenses. The MIT licence is considered to be a permissive open source license that at the same time does not limit the potential for commercial use.

In addition to the underlying application stack a number of additional libraries were used to develop the Skills Checker Tool client application. These libraries provided additional functionality such as video playback and file download/ saving. All of these libraries are made available through the MIT license and were documented in the technical handover report and summarised in the table below (libraries that are included as dependencies are also listed and denoted by a -). Similar to the underlying application stack (Angular and Loopback), these libraries are not bundled with the source code for the Skills Checker Tool. They are referenced in the configuration files and downloaded by any developer who might check out the source code from version control and build the project in the future.

Technology	License
angular-bootstrap-md	MIT License
- fortawesome/fontawesome-free	MIT License
- animate.css	MIT License
- chart.js	MIT License
- hammer.js	MIT License
videogular2	MIT License
file-saver	MIT License
ng5-slider	MIT License
normalize.css	MIT License

Similarly, the underlying database that stores the Skills Checker Tool's data is not distributed with the tool itself. However, we chose to make use of a popular and freely available database solution that any future developer or provider of the Skills Checker Tool could download and install without additional costs. We chose to use the PostgreSQL relational database for this purpose although providers are free to choose their preferred alternative open source solution such as MySQL or their preferred commercial solution such as Microsoft SQL Server or Oracle Database. In fact the Loopback technology used to build the server side component of the Skills Checker Tool supports a wide range of database technologies and the model driven architectural approach used means that it is relatively agnostic to the underlying data storage technology. In the future the Skills Checker Tool could be easily moved to a platform that doesn't even make use of a relational database at all.

One feature of the Skills Checker Tool that does rely on a commercial 3rd party service is the read aloud functionality. This functionality is provided by integrating with the service provided by ReadSpeaker. This functionality was requested by the project partners as part of the updates carried out to the app following on from the pilot evaluations and usability testing. Any organisation that wants to make use of this functionality in the future will need to sign up and pay for this service before then reconfiguring the Skills Checker Tool to use the appropriate settings to access the tool (ID, language, voice).

When integrating this functionality into the tool we provided a facility to easily enable/disable this feature through the application's configuration file at build time. This configuration functionality was used as part of the Maltese translation of the app. It was necessary to disable the read aloud feature as ReadSpeaker does not support the Maltese language. The flexibility of this configuration support also means that the tool can be quickly modified to meet the needs of different stakeholders. Different voices or languages can be selected without the need to make multiple changes across the application (As part of the CITO project the Alice voice was used in Ireland, the Hugh voice was used for Maltese English and Lykke was used in Norway). These voices do however have to be enabled by ReadSpeaker for the specific license.

In discussing the ReadSpeaker provided functionality it is important to note that as this functionality is service based, no ReadSpeaker code is actually included in the Skills Checker Tool. The integration works by loading the ReadSpeaker code from their server (or content delivery network) at runtime (ie. in the user's web browser).

GitHub

GitHub is a for-profit company that offers a cloud-based Git repository hosting service. Essentially, it makes it a lot easier for individuals and teams to use Git for version control and collaboration. The CITO Project team chose to host the source code, technical documentation and files for the project on GitHub. The interface is user-friendly enough so even novice coders can take advantage of Git. Without GitHub, using Git generally requires a bit more technical savvy and use of the command line. Additionally, anyone can sign up and host a public code repository for free.

The following can be accessed via the CITO Project's repositories on GitHub:

- Source Code
- Scripts for the animations
- Translation files
- Subtitle files
- Narration files
- Technical documentation including license files, README files
- MP4 files for the animation content
- Project Wiki

Here are the links for the two GitHub repositories foe the project:

- CITO Project: skills-checker-client-app
- CITO Project: skills-checker-api

Animation Content

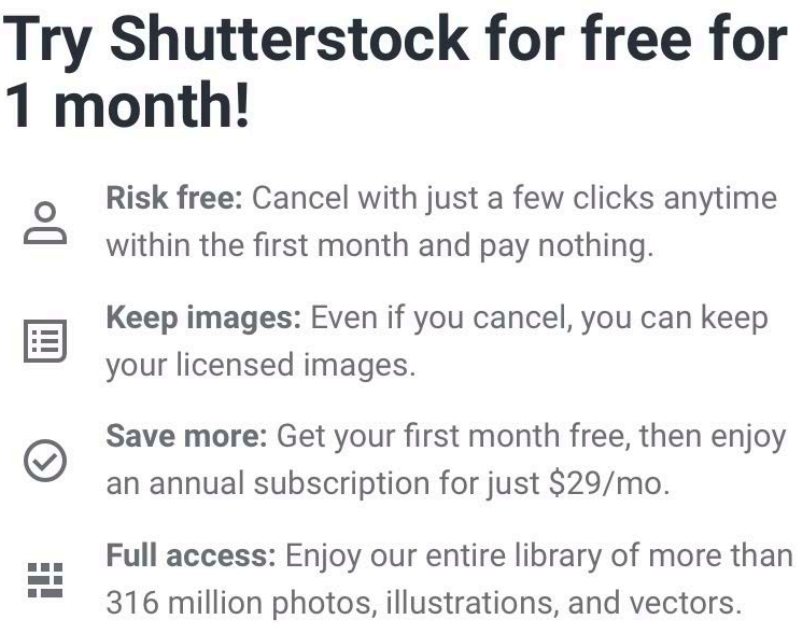
The MP4 files of the completed animations are available in the three different languages to download from GitHub. To aid future localisation and adaptation please see below for the specific links to the design assets that were used to create the animated content. These assets are free to download and will aid the reproduction of the animated content alongside the storyboard files. The following assets are free to download and to use without attribution from Sketch App:

- Backgrounds
- School bus
- Icons; cloud, sun
- Devices/Business
- Car
- Fence
- Phone/keyboard

The following assets are available for free from Freepik if you download Freepik resources as a free user and include the attribution line "Designed by Freepik:

- Italy Map – Social Goal, Animation 2
- Bus – Social Goal, Animation 1
- Books – Social Goal, Animation 3; Education Goal, Animation 4
- Spaghetti – Social Goal, Animation 3
- Plants – Social Goal, Animation 4
- Thought Bubble – Career Goal, Animation 1 and 2; Education Goal, Animation 2 and 3

There are seven remaining assets that are licensed via Shutterstock. Shutterstock provides a 1 month free trial, which gives you access and keep up to 10 assets:



Please see below for the links to the Shuttershock assets and where these assets are located in the animations:

- Man in wheelchair – Education Goal, Animation 2
- Old woman – Social Goal, Animation 3
- Woman tourist – Career Goal, Animation 1
- Old man – Social Goal, Animation 1; Personal Goal, Animation 4; Career Goal, Animation 4
- Boy – Personal Goal, Animation 1
- Woman – Education Goal, Animation 1 and 3; Career Goal, Animation 4; Personal Goal, Animation 1
- Man – Education Goal, Animation 4; Personal Goal, Animation 3; Social Goal, Animation 2; Social Goal, Animation 4; Career Goal, Animation 2 and 3
- The avatars (Maria, Frank, Petra).

Resources

The resources listed below will help support the animation localisation and adaptation process:

- Animation storyboard files
- Animation audio files
- Animation subtitle files
- MP4 animation files (Norwegian, English, Maltese, and Simplified English)

Licenses

Licenses

The table below summarises the 3 main open source licenses under which 3rd party software components used to implement and/or run the SkillsCheck tool are released.

License	Description
MIT	Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
Apache 2.0	Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0
OBSD	Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License. The BSD Zero Clause license goes further than the BSD 2-Clause license to allow you unlimited freedom with the software without requirements to include the copyright notice, license text, or disclaimer in either source or binary forms.

For further information see:
https://citoproject.eu

Contact the CITO Project team:

cito@gov.mt

Waterford Institute of Technology

checkin takeoff

Co-funded by the Erasmus+ Programme of the European Union

NALA
National Adult Literacy Agency
Áisíneacht Náisiúnta Litearthachta do Aosaigh

etbi
Education & Training Boards Ireland
Boird Oideachais & Oiliúna Éireann