0.1 Functionalities

The subsections below describe the functionalities of UltraCast broken down into 5 major functionality groupings. Each functionality is directly connected to a User Story, defined in the Proposal, as well as a project objective if it addresses it. The colour schema defined in Table 1 was used for functionalities in Tables 2 to 6.

Table 1: Functionality Table Colour Schema

Colour	Meaning
	Functionality relates to project objective
	Functionality not specified in project objective
	As above, but functionality is novel

0.1.1 Viewing and Searching

The viewing and searching functionalities are described in Table 2 below.

Table 2: Viewing and Searching Functionality Mapping

Story ID	Functionality	Project Objective
UL-2 UL-3	Use keywords to search for podcasts, return list of podcasts (See UL-4 for format) View the total number of subscribers for each podcast returned from a search	Listeners must be able to search for podcasts that interest them by keywords, resulting in a list of matching podcast titles, where the total number of subscriptions on the UltraCast platform (function described later) for each podcast is shown next to the title
UL-4	View the title, description, author details and list of episodes for a podcast	Listeners must be able to select a podcast show from returned search results to view its full details, including its title, description, any author details that exist, as well as a list of episodes for the show
UL-14	Login as specific user	-
UL-24	View a title, length, upload date for episodes	-
UL-29	Save search as a "Stream"	-
UL-41	Sign up as user	-

0.1.2 Playing Podcast Episode

The playing podcast episode functionalities are described in Table 3 below.

Table 3: Playing Episode Functionality Mapping

Story ID	Functionality	Project Objective
UL-5 UL-6	Play episodes Once episode starts being played it is marked as played	Listeners must be able to play a selected episode within a podcast show, and once that episode starts being played, the listener must be able to also clearly see this episode marked as "Played"
UL-18	Pause episode that is playing	-
UL-19	Adjust playback volume	-
UL-20	Skip to next episode, previous episode and start of current episode	-
UL-21	Jump to a point in an episode	-
UL-22	Adjust playback speed	-
UL-23	Auto-play episodes in a podcast (after added to playlist)	-
UL-26	"Bookmark" a point in an episode with a title and description	-

0.1.3 Recommendation and Following

The recommendation and following functionalities are described in Table 4 below.

Table 4: Recommendation and Following Functionality Mapping

Story ID	Functionality	Project Objective
UL-10 UL-11	View episode history Episode history is sorted by most recent to least recent	Listeners must be able to see a history of the podcast episodes that they have played, sorted in order from most recently played to least recently played
UL-12 UL-13	Podcast recommendations are based on: Existing subscriptions recently played episodes and past searches A "recommended" panel shows recommended podcasts	UltraCast must be able to recommend new podcast shows to a listener based on at least information about the podcast shows they are subscribed to, podcast episodes they have recently played, and their past podcast searches
UL-18	Follow users, view their listen history	-

0.1.4 Creator Mode

The creator functionalities are described in Table 5 below.

Table 5: Creator Mode Functionality Mapping

Story ID	Functionality	Project Objective
UL-15 UL-16	Create podcasts and episodes Delete podcasts and episodes	-
UL-17	Update podcasts and episodes	-
UL-27	Access to podcast and episode viewer metrics	-

0.1.5 Subscribe

The subscribe functionalities are described in Table 6 below.

Table 6: Subscribe Functionality Mapping

Story ID	Functionality	Project Objective
UL-7 UL-8	Subscribe to podcasts Unsubscribe to podcasts	Listeners must be able to see a history of the podcast episodes that they have played, sorted in order from most recently played to least recently played
UL-9	User receives notification for each new episode in a podcast they are subscribed to	Listeners must be notified by the platform when a new episode for a show they are subscribed appear
UL-30	The latest episode for each subscribed podcast is linked in the "Subscriptions" page	Listeners must be able to see the latest episode available for each show that they subscribed to in a "Podcast Subscription Preview" panel

0.2 System

0.2.1 GraphQL

GraphQL is a query language for APIs and a runtime for fulfilling these queries. It provides a complete and understandable description of the data in your API, with a type system to ensure that apps can only make logical requests - providing clear and helpful errors when necessary. This works to improve the reliability of any solution built upon it.

GraphQL queries also return predictable results by allowing clients to define the structure of the data required; the queries supply exactly what a client asks for and nothing more. This ensures the speed and stability of the apps built using GraphQL, since the app itself controls the data it receives instead of the server. GraphQL also comes with a powerful developer GUI that allows you to visualise exactly what data your API has available. It also highlights potential errors in test queries and leverages code intelligence to make helpful suggestions on how to fix them.

GraphQL is naturally scalable, as any type or amount of data can be contained inside a single query. New queries can be developed on the fly and existing queries can be modified all without breaking existing functionality.

0.2.2 MongoDB

MongoDB is a document database designed to be used in the development of scalable applications using agile methodologies. MongoDB naturally supports rapid iterative development, which fits the nature of this project perfectly. It scales to high levels of read and write traffic and to a massive database size, and can easily evolve the type of deployment as necessary; removing roadblocks to development if the structure of our solution changes. As MongoDB relies on JSON files to store data, the structure of the information is under the control of the developer. Developers can adjust and reformat the database as the application evolves with ease - further supporting the RAD approach.