Requirements		**Type of Test** (I: Integration,Testing, S: System Functionality Testing, O: Operational Acceptance Testing, UN: Unit Testing, US: Usability Testing, A: Acceptance Testing)	**Pass or Fail** (P: Pass, F: Fail)	**Contributor** (PM: Patrick Ma, KK: Kiyoon Kim, TD: Tanner Dyck, SK: Sten Korver, GG: Gus Goerzen)
Functional Requirements				
	1 - Users			
1.1	Users can create an InstaQuiz account with instructor, or student privileges using their email address and a custom password.			
1.2	Users can login to their InstaQuiz accounts using their email address and password.			
1.3	Users can logout by clicking a constantly displayed button in the top right of the web-page.			
1.4	Instructors can create a new course module, this automatically enrolls them in the course, as the instructor.			
1.5	Instructors can delete any course module they have personally created.			
1.6	Users can search for a course module by title or instructor			
1.7	Students can enroll in any course module that they are not currently enrolled in.			
1.8	Students can unenroll from a course they are currently enrolled in.			
1.9	Instructors can start a live course session in any courses they have created.			
1.10	If an instructor has any ongoing live sessions, they can end them whenever, as long as no polling windows are active.			
1.11	Users can join a live course session, provided that the instructor has started it already.			
1.12	Instructors can create a storage bank of custom questions with corresponding multiple choice answer sets.			
1.13	Instructors can start a polling window during a live session as long as no other polls are already in progress.			
1.14	Instructors can end their live polling window and declare the correct answer to the question.			
1.15	Users can answer a question by clicking on the corresponding answer listed beneath the question text. A user can change their answer during the polling window.			
1.16	Instructors can choose to display a summary of the question, (correct answer, how many votes each answer got, how many total responses, etc.) or to keep it hidden from the class. Instructors will be able to see this summary regardless after they end a polling window.			
1.17	Students can see how many questions they have answered right for a particular course. The total questions asked for the course will also be shown so the student will see a comprehensive score, something like 16/21.			
1.18	For each course they are enrolled in, students will be able to see their current score, attendance record, and response history throughout previously attended live sessions.			
1.19	For each course they have created, instructors will have access to their student’s grades, attendance records, and response history.			
1.20	Students and instructors can view the InstaQuiz history for a course which will show all past polls and questions with the corresponding correct answers.			
	2 - Interface			
2.1	Students have an overview page which displays each course they are enrolled in, and gives them the option to join a class, or unenroll from one.			

2.2	Students can access course pages from the overview page where they will then be able to view their course statistics and join live sessions that are currently running.			
2.3	During a polling window, the question statistics are displayed and updated in real-time. (How many students have answered/ not answered)			
	3 - Other Functional Requirements			
3.1	InstaQuiz will keep record of all the courses that have been created by instructors and store data about them such as their title, name of instructor, and past poll questions and answers.			
Non-Functional Requirements				
	1 - Performance			
1.1	Web service must update in real-time, users are given varying time windows to submit responses. We must minimize response times to and from the main client.			
1.2	Web service must be able to handle large amounts of traffic. As an international education tool, thousands of academics may be on the site at one time. Live sessions must be able to support over 300 connected clients at a time.			
	2 - Security			
2.1	Emails and passwords must be hashed before being stored. Verification will be done through hashing entries and comparing with hashes stored.			