[ ] = inferred requirement/nice to have

WEBSITE

Landing page: login

- hash password for secure transmission

- store session data while logged in ~~(JS: sessionStorage/localStorage)~~

- logout option on all other pages deletes session & redirects to landing page

[ check session data for uncommitted edits and notify before logging out ]

View all available quizzes

- if edit permission, options available to create/delete quiz

[ search by title fragment ]

Select quiz from list to view questions

[ list questions ordered by index ]

- if edit permission, options available to create/edit/delete question

[ commit changes via 'save' button, [store uncommitted changes in session] ]

[ new questions should include 3-5 answers when created ]

- able to insert created question anywhere in the list [ and modify order manually ]

- insert/delete other than at end requires renumbering questions

If edit or view permission, select question from quiz to view answers

[ list answers ordered by index [under the question, within the list of questions] ]

- if edit permission, options available to create/edit/delete answers

[ commit changes via 'save' button, [store uncommitted changes in session] ]

[ able to insert created answer anywhere in the list [and modify order manually] ]

- insert/delete other than at end requires re-indexing answers

[ for convenience, if edit permission, also allow editing question while viewing answers ]

DATABASE - postgres

Table: ref\_permissions (id INTEGER PK, description TEXT)

Table: user\_details (user\_name TEXT PK, user\_password TEXT, user\_permission INTEGER REFERENCES ref\_permissions.id)

Table: quiz (id SERIAL PK, title TEXT)

Table: question (id SERIAL UNIQUE, quiz\_id INTEGER references quiz.id, question\_index INTEGER, question\_text TEXT, PK(quiz\_id, index))

Table: answer (id SERIAL UNIQUE, question\_id INTEGER references question.id, answer\_index CHAR(1), answer\_text TEXT, [is\_correct BOOLEAN,] PK(question\_id, index))

Prepopulate user\_permissions in database setup script

Produce script for user detail input, separate from database setup script

[ consider making answer.index an enum type (A-E) or adding validation in an ingest function as the definition allows 3-5 answers on a question; document for maintainability if implemented]

[ all fields in all tables NOT NULL ]

[ hashing of user.user\_password will not be handled by the database, the hashed value will be received and stored unmodified ]

[ define stored procs for question & answer modifying actions, to include re-indexing (PK constraints are a safety net to prevent duplication of indices) ]

[ checking user permissions will be the responsibility of the API, not the database ]

API - Java

logIn(userName, password)

getQuizzes()

getQuestions(quizId)

getAnswers(questionId)

createQuiz()

createQuestion(quizId, questionText, answerTexts...)

createAnswer(questionId, text)

updateQuestion(questionId, newText)

updateAnswer(answerId, newText)

deleteQuiz(quizId)

deleteQuestion(questionId)

deleteAnswer(answerId)

Need to consider how best to verify user status on each call - this is achieved by inspecting Spring's SecurityContext and passing the permission information to the view for conditional rendering of HTML elements. Spring Security ensures users can only view the login page until they sign in with valid credentials.