

Test

Unit tests

```
PASS middleware/middleware.spec.ts
middleware
  encode payload
    ✓ it returns an encoded payload (2 ms)
  decode payload
    ✓ returns decoded payload (1 ms)
    ✓ returns type invalid-token if there is an error
  check expiration status
    ✓ returns active for a token that is less than 15mins old (1 ms)
    ✓ returns expired for a token that is more than 15mins old

PASS services/login-service.spec.ts
login-service
  verifyUsername
    ✓ returns true if username is in the database (44 ms)
    ✓ returns undefined if username is not the database (19 ms)
  hashPassword
    ✓ returns a string that differs to the password (15 ms)
  verifyPassword
    ✓ returns true when given a valid username and password (12 ms)
    ✓ returns undefined if username and password (13 ms)
  getPayload
    ✓ returns a session when given a valid username and hashed password (19 ms)

PASS services/quiz-service.spec.ts
quiz-service
  getQuizzes
    ✓ returns a list of quizzes (35 ms)
  verifyQuizId
    ✓ returns true if quizId is in the database (19 ms)
    ✓ returns undefined if quizId is not the database (12 ms)
  checkQuizName
    ✓ it returns true if quiz name does not exist (12 ms)
    ✓ returns undefined if quiz name already exists (10 ms)
  addQuiz
    ✓ returns quiz id of new quiz (12 ms)
  deleteQuiz
    ✓ deletes quiz and corresponding questions and answers and returns undefined (43 ms)
  getQuiz
    ✓ returns a quiz of the correct structure (19 ms)
```

```

PASS services/answer-service.spec.ts
answer-service
  checkNumberOfAnswersLessThanFive
    ✓ returns true if number of answers is less than five (47 ms)
    ✓ returns undefined if the number of answers is more than 4 (15 ms)
  checkNumberOfAnswersMoreThanThree
    ✓ returns true if number of answers is more than three (13 ms)
    ✓ returns undefined if the number of answers is more than 4 (11 ms)
  addAnswer
    ✓ returns answer id of new answer (15 ms)
  checkAnswerIsNotCorrect
    ✓ returns true if the answer is not correct (14 ms)
    ✓ returns undefined if the answer is correct (15 ms)
  deleteAnswer
    ✓ deletes answer and returns undefined (26 ms)
  updateAnswer
    ✓ updates answer and returns undefined (23 ms)

PASS services/question-service.spec.ts
question-service
  checkNumberOfQuestions
    ✓ returns true if number of questions is less than 26 (17 ms)
    ✓ returns undefined if number of questions is 26 or more (8 ms)
  checkCorrectAnswerExists
    ✓ returns true if the correct answer exists (17 ms)
    ✓ returns undefined if correct answer does not exist (7 ms)
  addQuestion
    ✓ returns question id of new question (9 ms)
  deleteQuestion
    ✓ deletes question and corresponding answers and returns undefined (28 ms)
  updateQuestion
    ✓ updates answer and returns undefined (14 ms)
  updateCorrectAnswer
    ✓ updates correct answer and returns undefined (15 ms)

```

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	100	100	100	100	
middleware	100	100	100	100	
check-experiation-status.ts	100	100	100	100	
decode-payload.ts	100	100	100	100	
encode-payload.ts	100	100	100	100	
services	100	100	100	100	
answer-service.ts	100	100	100	100	
login-service.ts	100	100	100	100	
question-service.ts	100	100	100	100	
quiz-service.ts	100	100	100	100	

```

Test Suites: 5 passed, 5 total
Tests:       36 passed, 36 total
Snapshots:   0 total
Time:        3.307 s

```

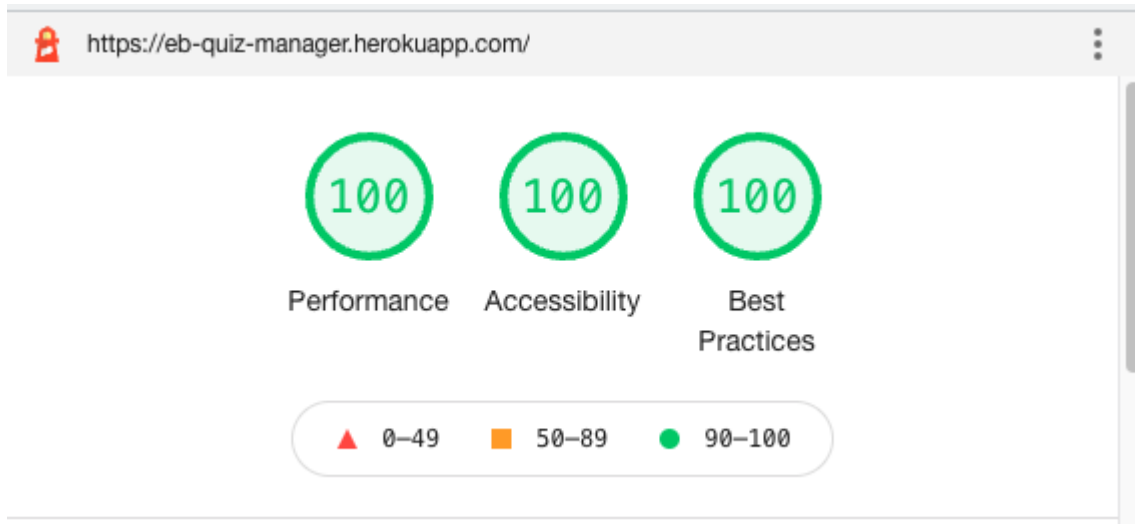
I used TDD whilst writing my sql queries and middleware to ensure that the code produced the expected result. I used a docker compose file to spin up different postgres instances containing the schema in order to test against a representative environment. Each query test suite runs against its own postgres instance to prevent interference between tests. E.g. The question test suite deletes a question, which is later relied on by the answer suite to add a question. All test suites are run on git commit to prevent bugs being introduced into the code.

Lighthouse testing

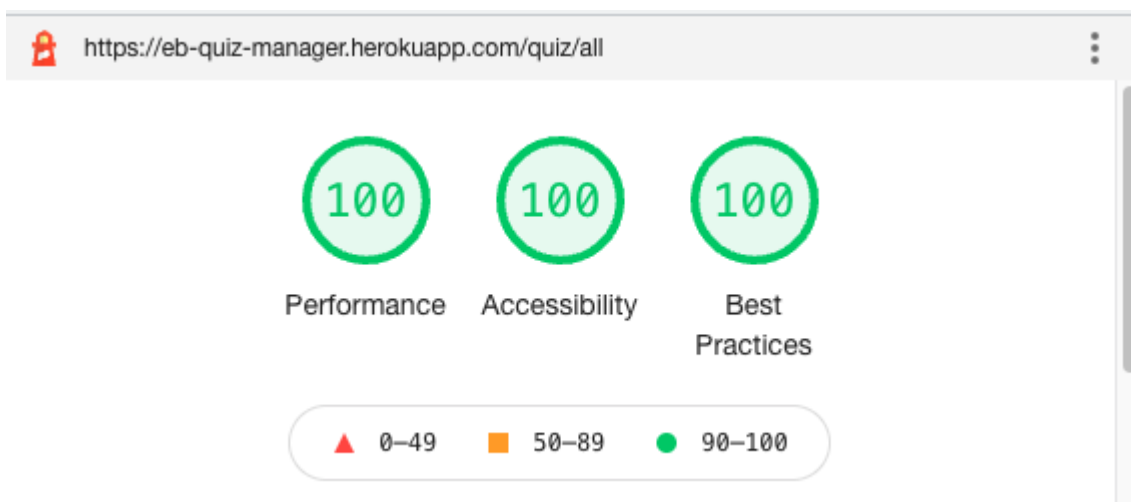
I used lighthouse report to check the sites performance, accessibility and best practise scores in order to help identify any issues.

The final results can be seen below.

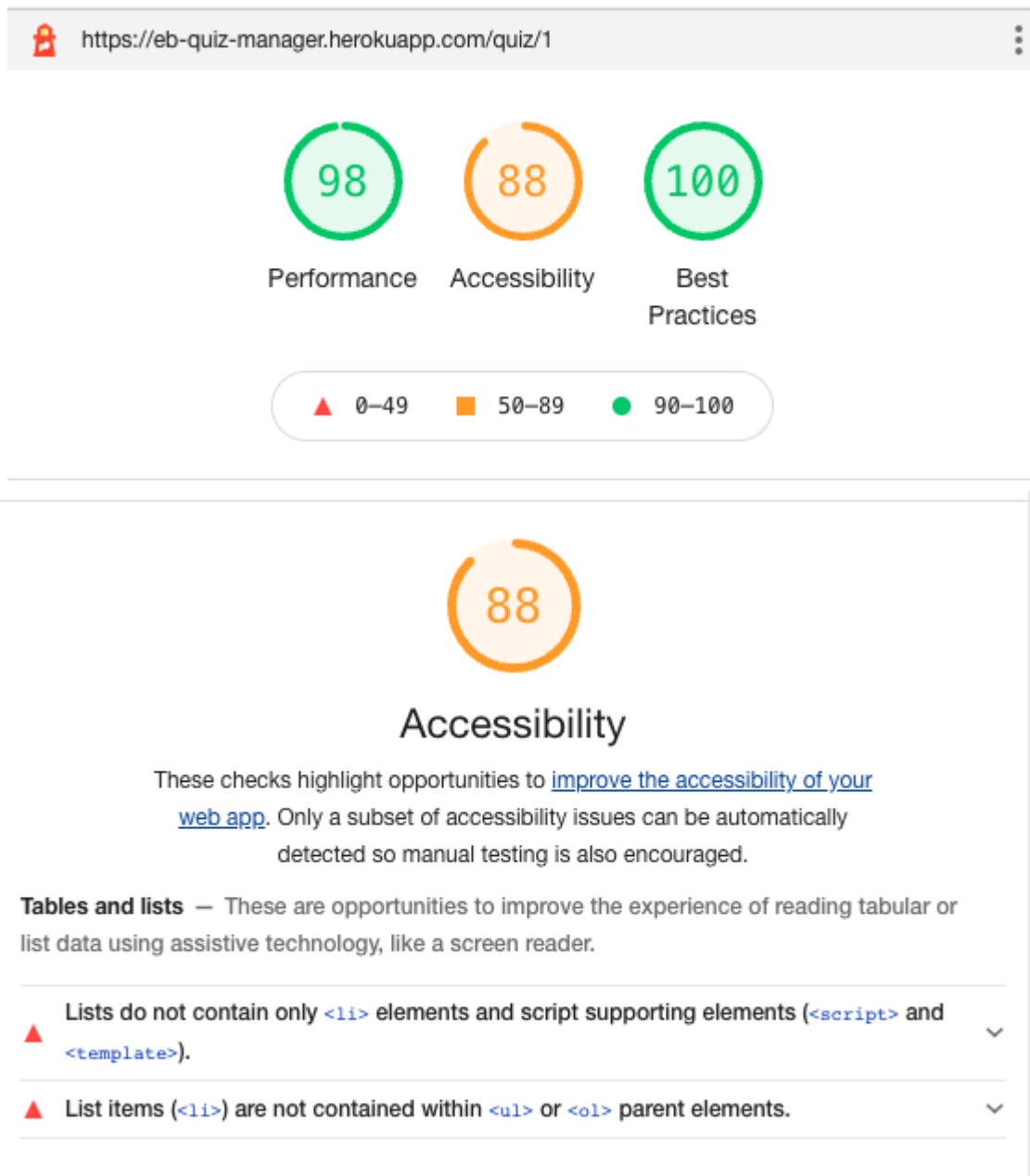
Login page '/'



View all page '/quiz/all'

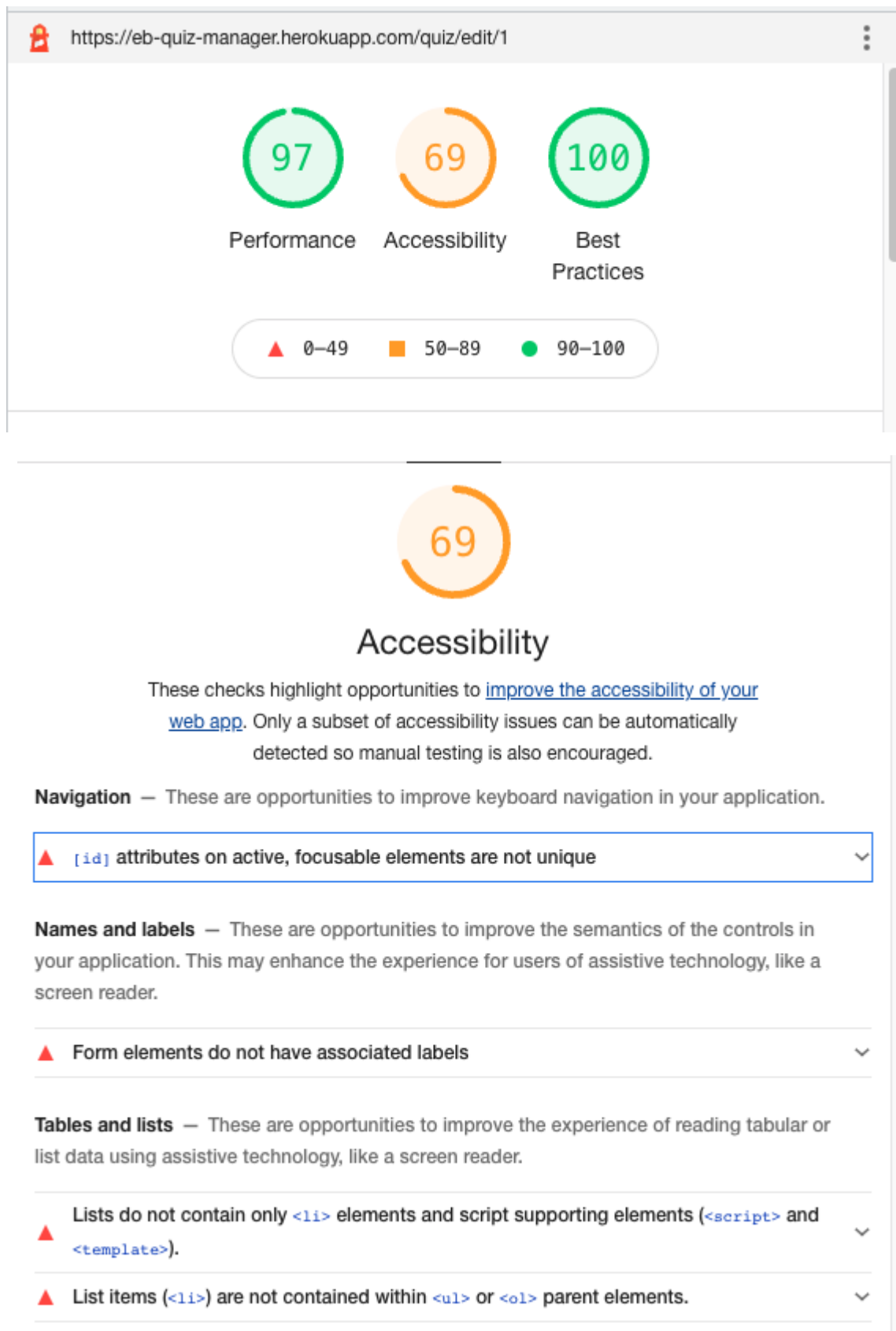


View quiz page '/quiz/:id'



To produce lettering of each question, each question is contained in a div. Each div has an `` element with type set to A inside it, and all question divs are contained inside of an `` element. The only other solution I could think of to solve this issue was writing a handlebars helper to work out alphabet letters of questions based on index. I, however, ran out of time to try this.

Edit quiz page '/quiz/edit/:id'

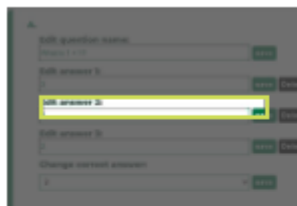


This page had the same issues as above alongside an issue with form labels and unique ids on elements.

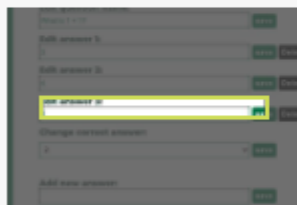
The form in question was the add question form, and upon looking at the code I could see that the form labels were present. I also checked this with voiceover for mac and was able to fill in the form and read the labels, however I admit that I still need to improve my use of screen readers and may have missed an anomaly from how most forms read.

The unique id issue, again was not present, which can be seen in lighthouse itself:

Failing Elements



`input#edit-answer-input-3.edit-input`

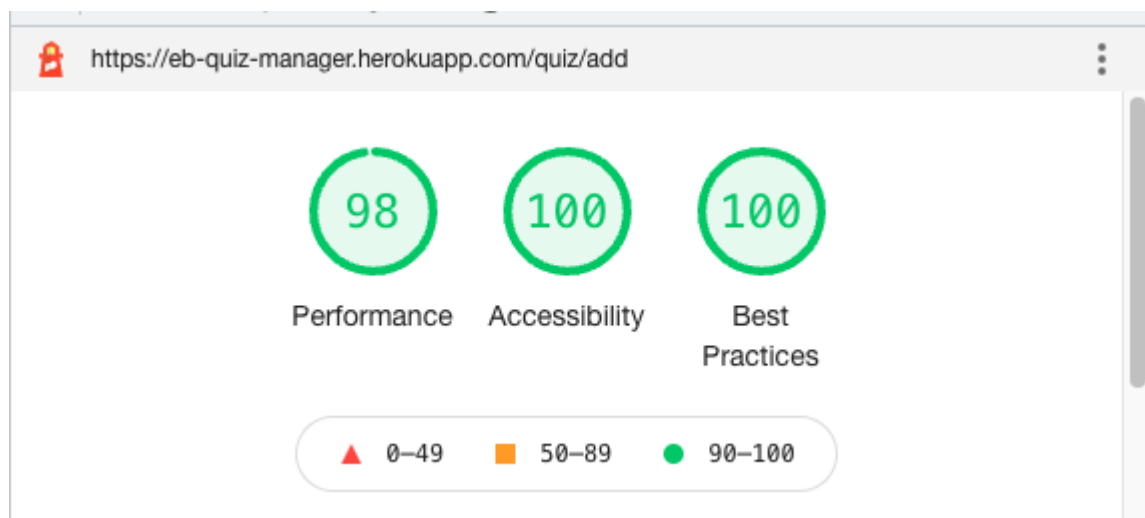


`input#edit-answer-input-4.edit-input`



`input#edit-answer-input-2.edit-input`

Create quiz page '/quiz/new'



Manual testing

The manual test case document can be found completed in the testing folder. The only failing test was that logged out users are redirected to the login page when they visit any url. When visiting a made up url they are redirected to the 404 page. This is not the ideal behaviour however it does not create any issues.