Blazor Web Documentation

BR20-2

2021

# Foreword

Thank you for granting me the opportunity to be a candidate to join the Research and Development team in Software Laboratory Center at Binus University. Thank you for everyone at Operation Division for leading me throughout the semester, everyone at Research and Development for choosing me to be a candidate, and everyone at Software Laboratory Center who has helped and accompanied me to reach the state I am currently at, especially my generation (20-2). I am grateful and will try my best at doing this test.

# Technologies

## ASP.NET Core 5.0

ASP.NET Core 5.0 is the latest web framework for developing web applications on the .NET platform by Microsoft. It is better from its predecessor, ASP.NET 4.x, as ASP.NET Core is much more performant, provides cross platform solution, and its latest UI technology, Blazor.

## ASP.NET Core Identity

ASP.NET Core Identity provides authentication and authorization scaffolding to ASP.NET Core projects with built-in supports for login register, social login, database migration, and more.

## Blazor (Server)

Blazor is part of ASP.NET to build interactive web UI with C#. There are two hosting models for Blazor, Blazor WebAssembly and Blazor Server. Blazor Server is used in this application to lighten up client’s load and prerender pages server-side before sending them to client.

## Entity Framework Core + SQL Server

Entity Framework Core is used as ORM in this application together with SQL Server to help in writing data-access code and optimizing database queries.

## MudBlazor

MudBlazor is material themed components made for blazor.

## Bootstrap 4.3.1

ASP.NET Core project template comes with Bootstrap 4.3.1 out of the box to enable rapid styling.

# Pages

All pages are placed under “Pages” folder.

## Home

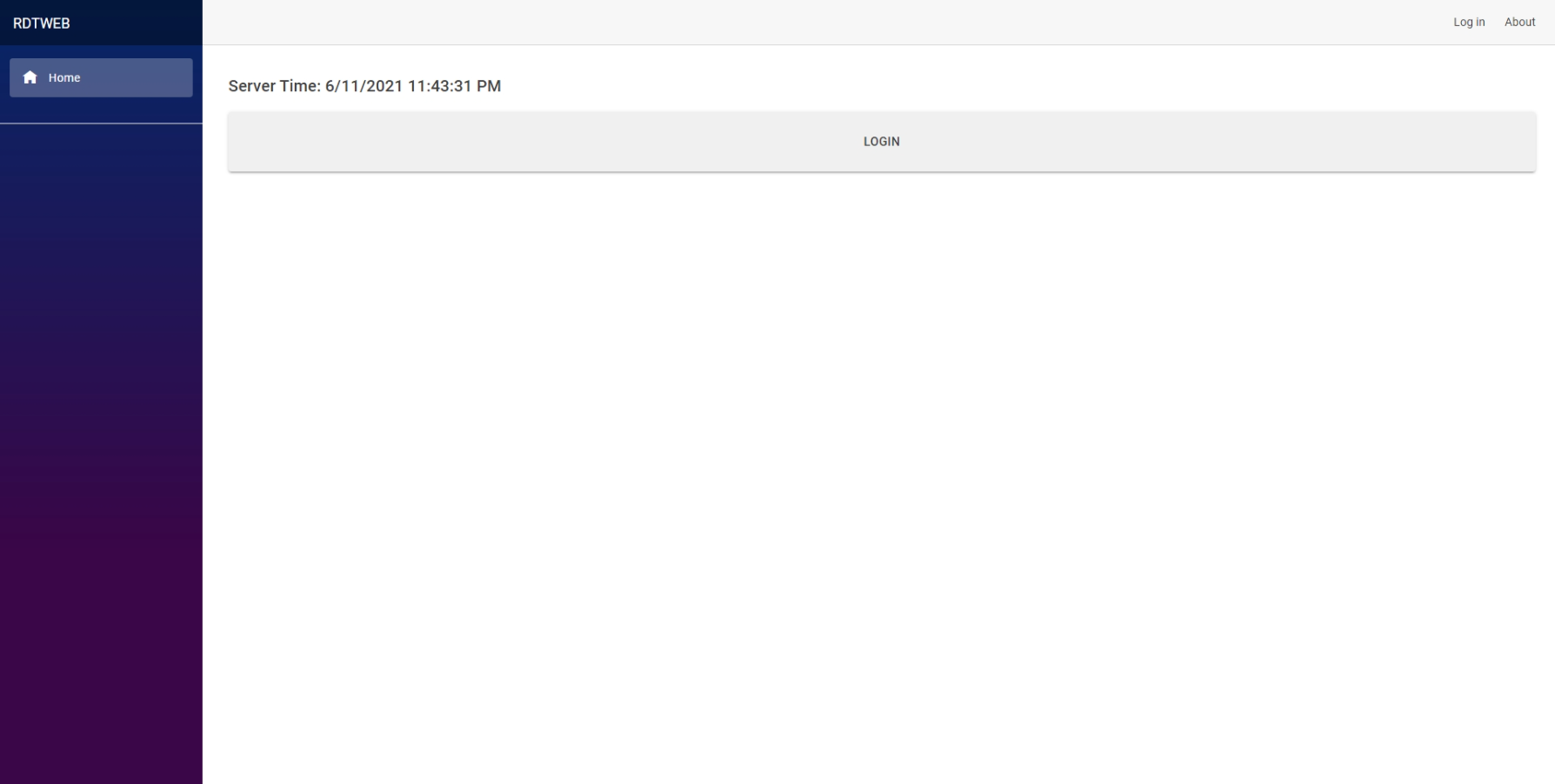


Figure . Home (guest).

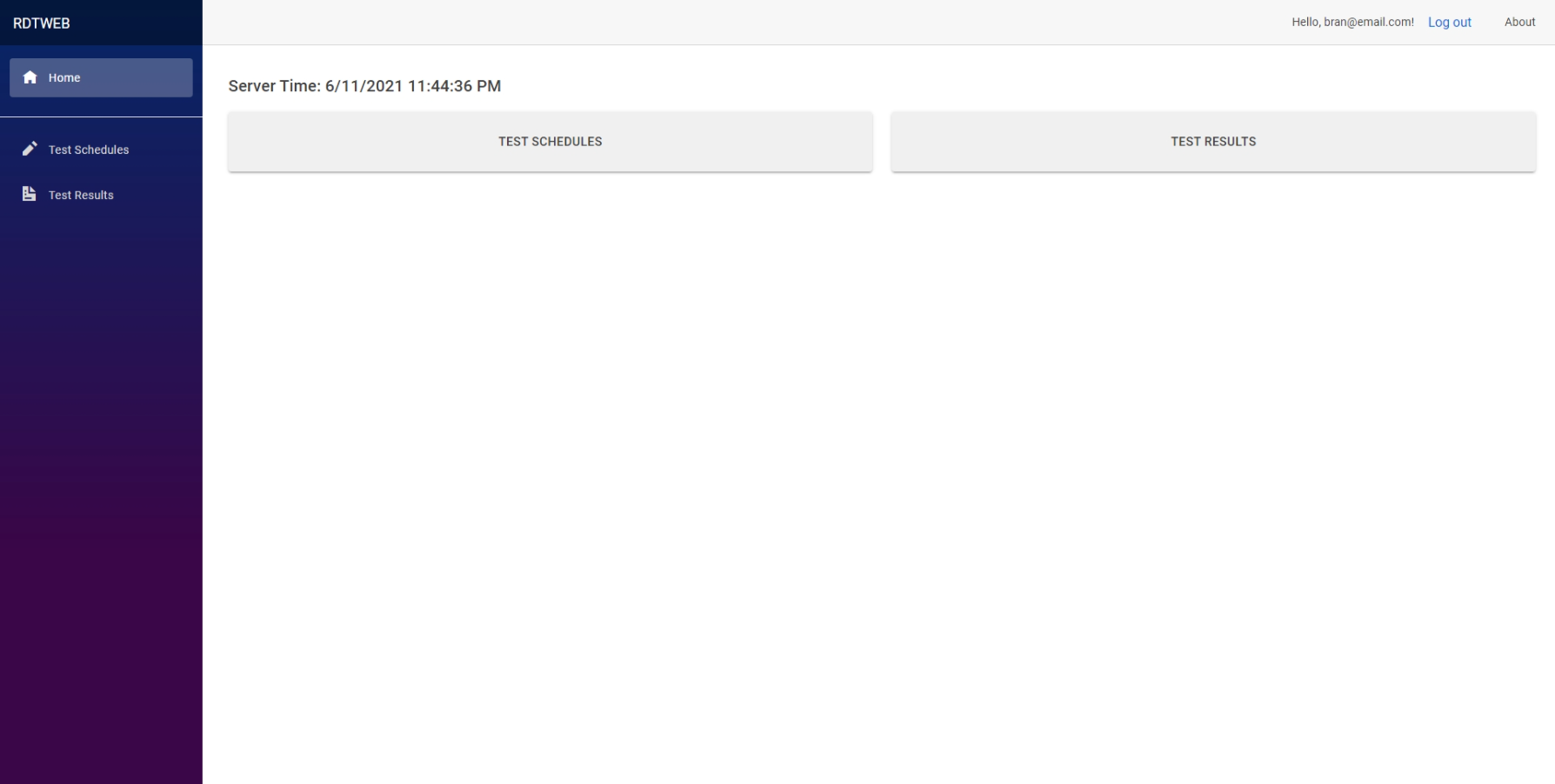


Figure . Home (participant).

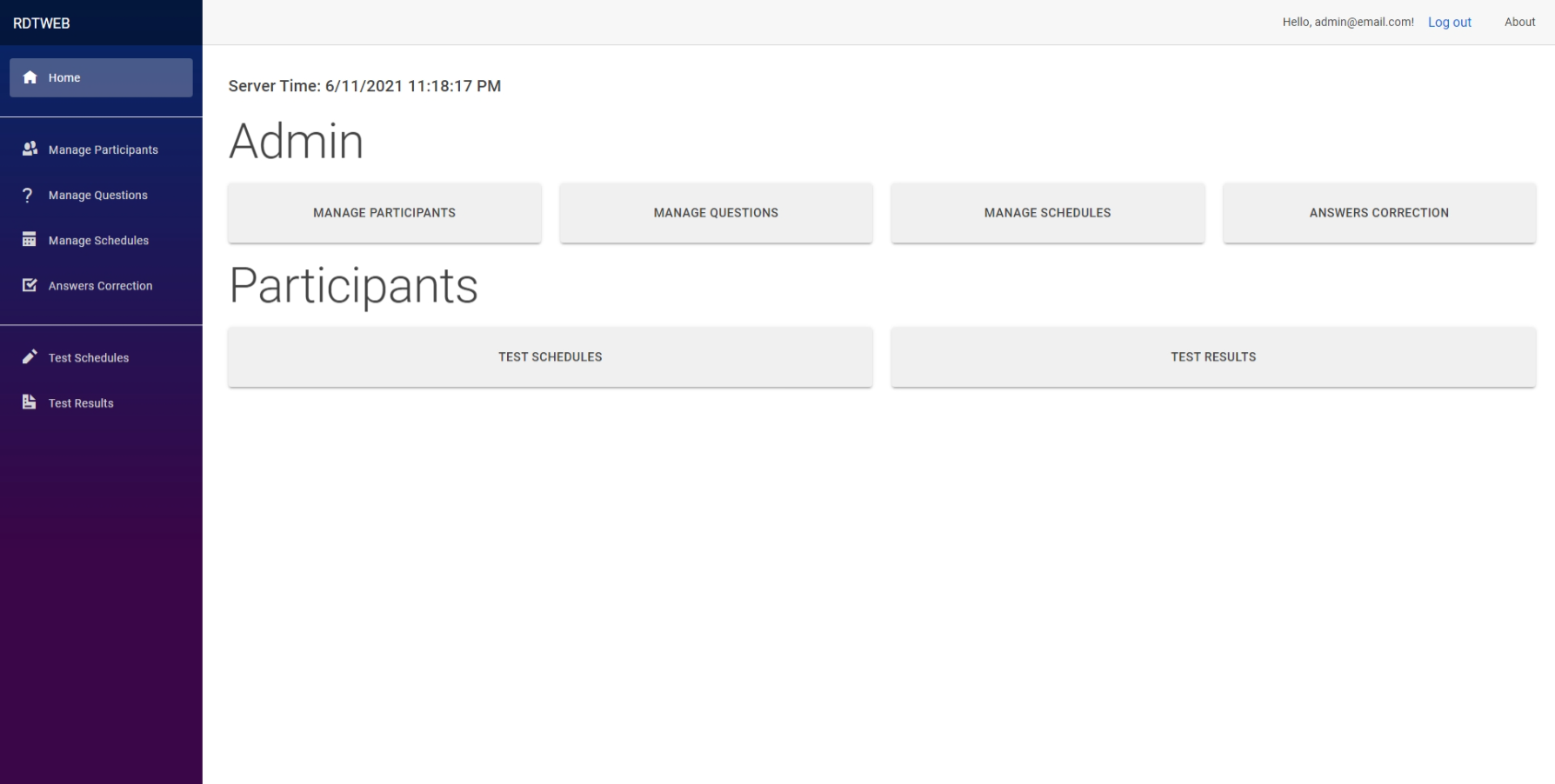


Figure 3. Home (admin).

User can go to navigate to every page from here. This page shows which pages that the user is authorized to visit.

## Admin

All pages under “Admin” folder are authorized for users with admin role only.

### Manage Participants

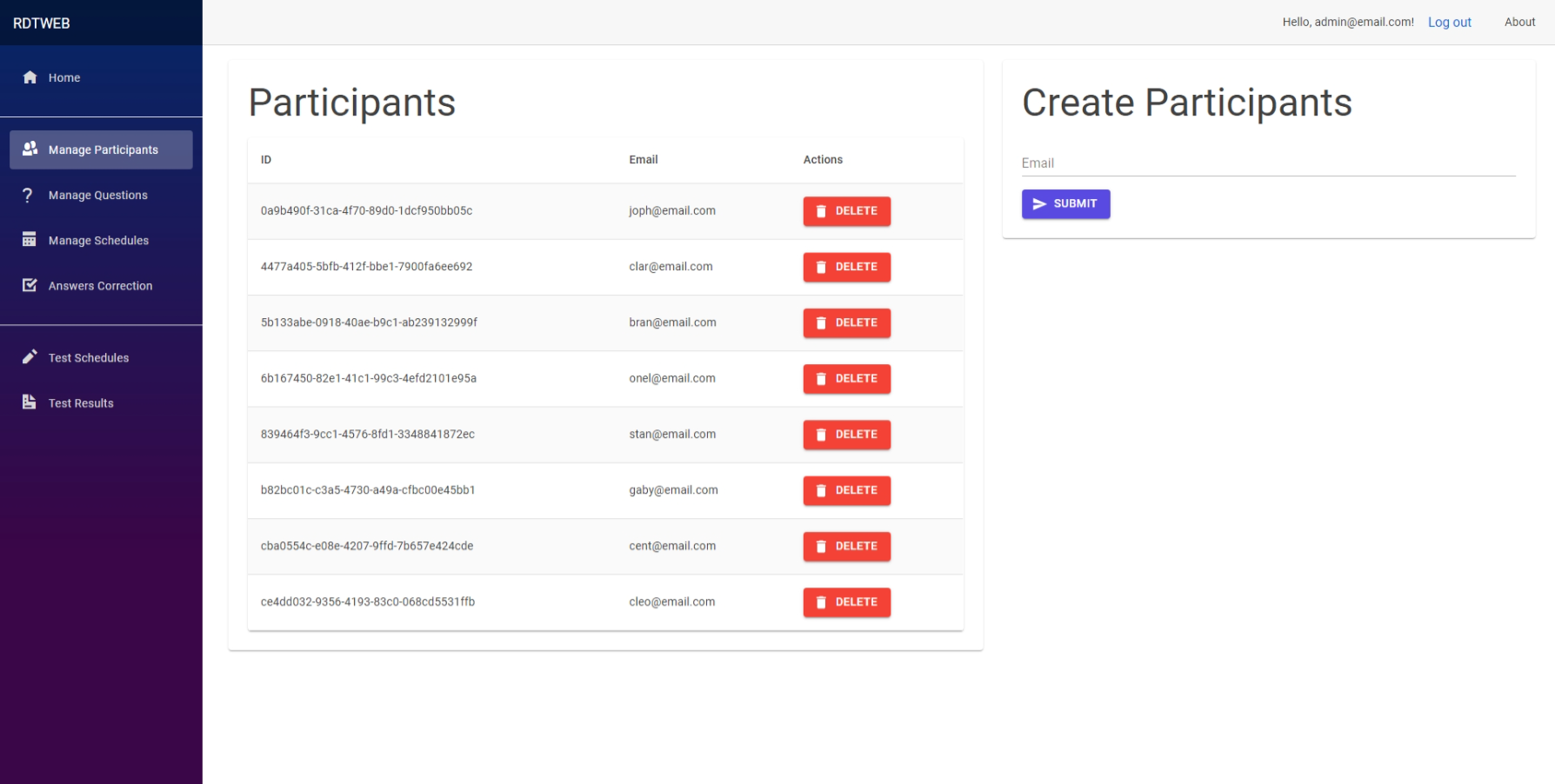


Figure 4. Manage participants.

Admin can view all participants, create, and delete participants in this page.

### Manage Questions

Admin can view all questions, view question detail, update and delete questions in this page. Manage Questions are divided into three pages: Index, Create, and View.

#### Index

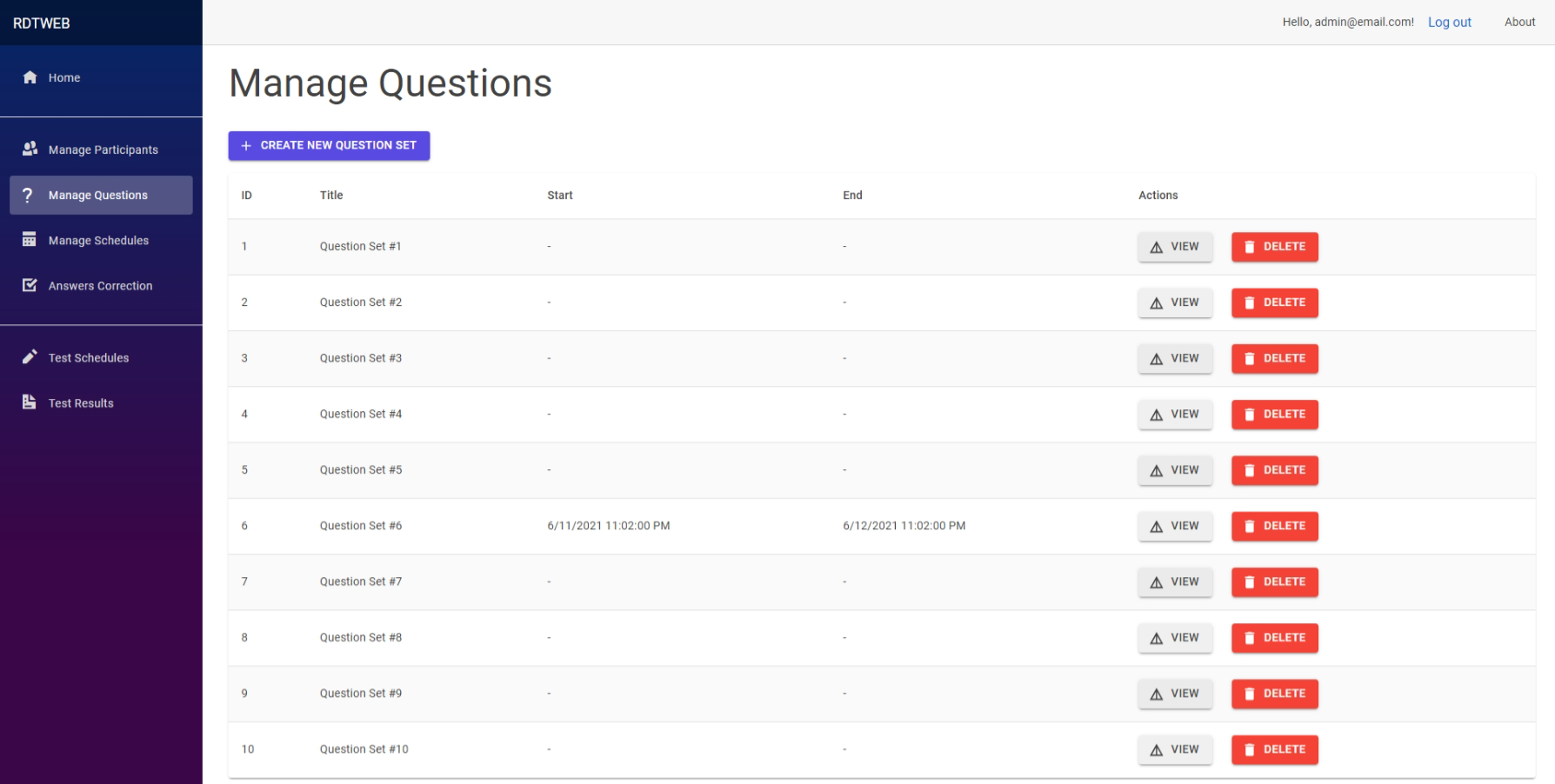


Figure 5. Manage questions index.

This page shows all question sets. Question sets can be deleted.

#### Create

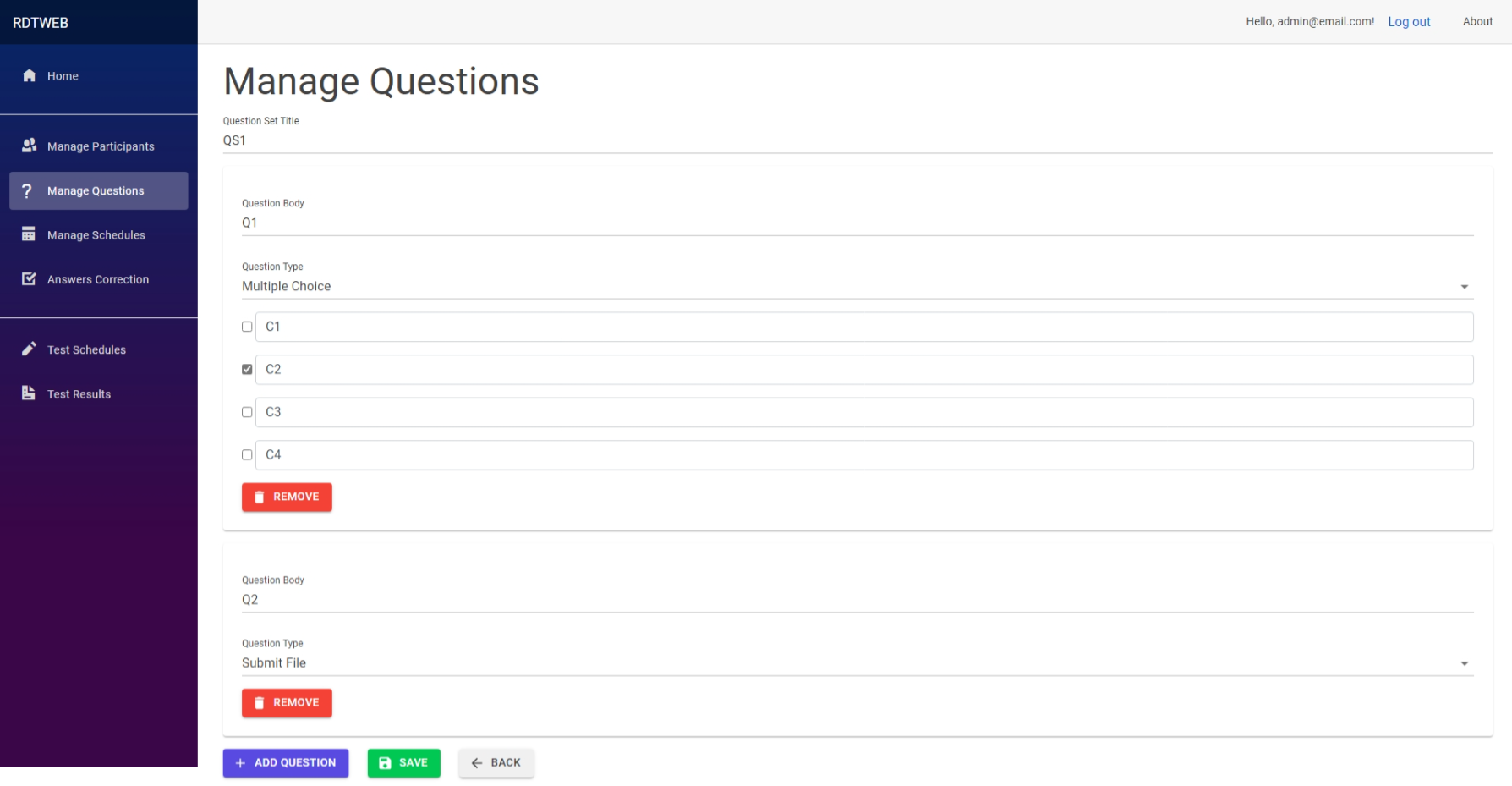


Figure 6. Create question set.

This page shows options for creating question set. Admin can also set question’s correct answer. Admin can choose any of the following 5 question types:

* Multiple Choice  
  User can choose one answer out of four possible answers.
* Boolean (true or false)  
  User can choose one answer out of two possible answers.
* Multiple Answer List  
  User can choose one answer out of indefinite possible answers. Admin can indefinite amount of possible answers.
* Essay  
  User can write multiline string with auto-save feature with delay of 500ms.
* Submit File  
  User can upload any file.

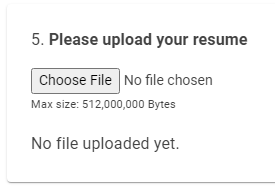


Figure . File upload (no file).

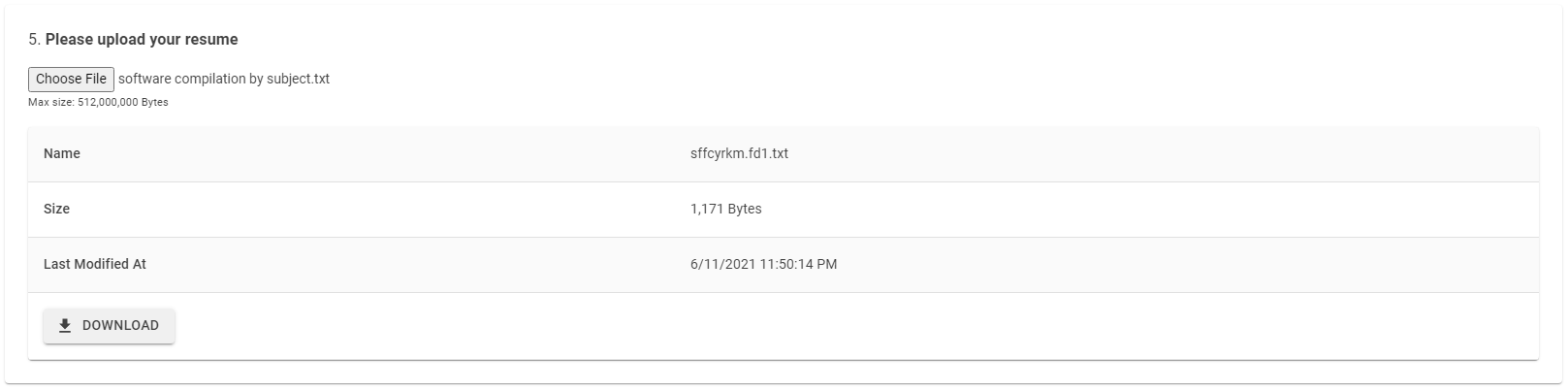


Figure . File upload (uploaded).

#### View

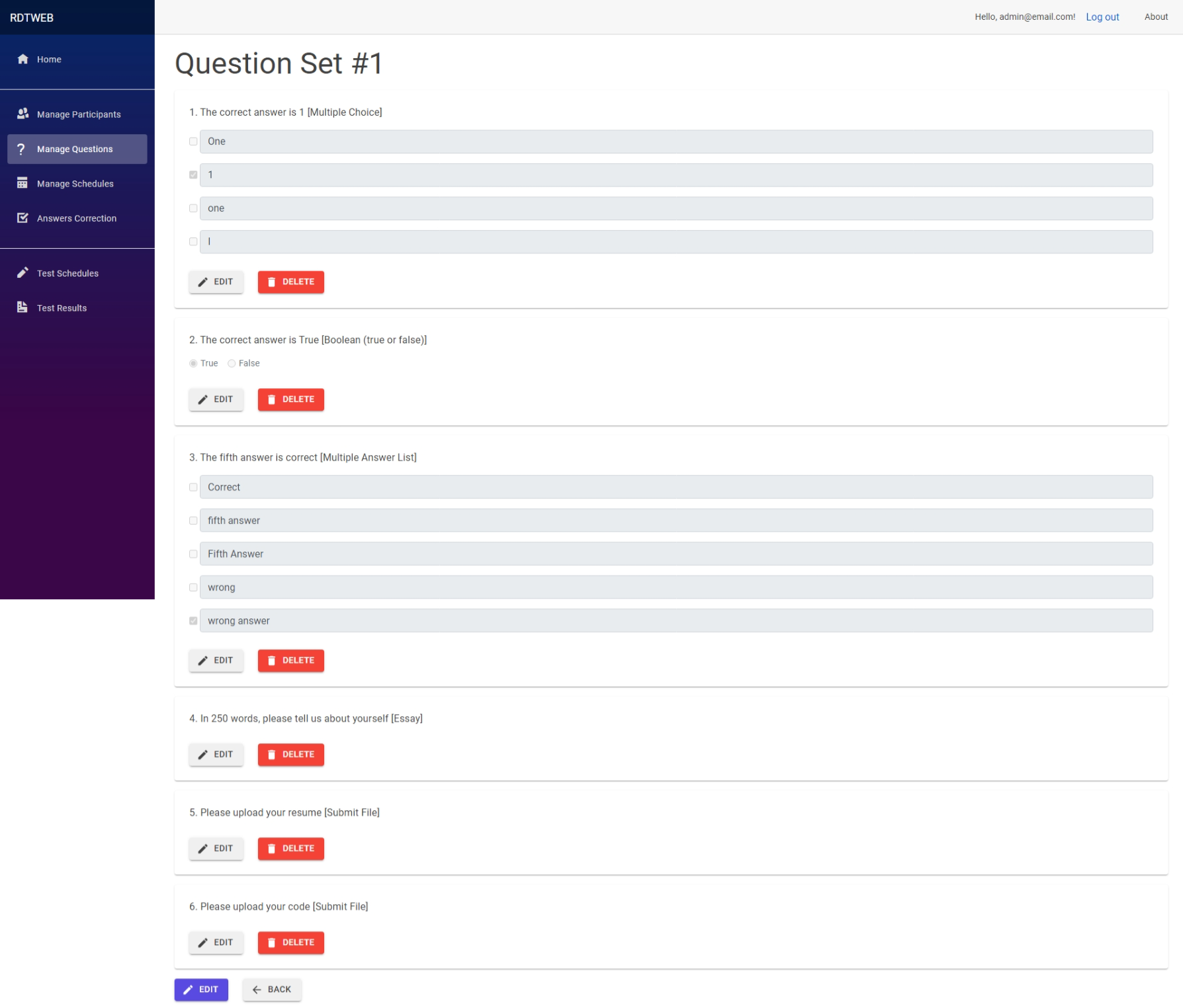


Figure 9. View question set detail.

Admin can view all questions in a question set, update its type, and correct answer, and delete questions.

### Manage Schedules

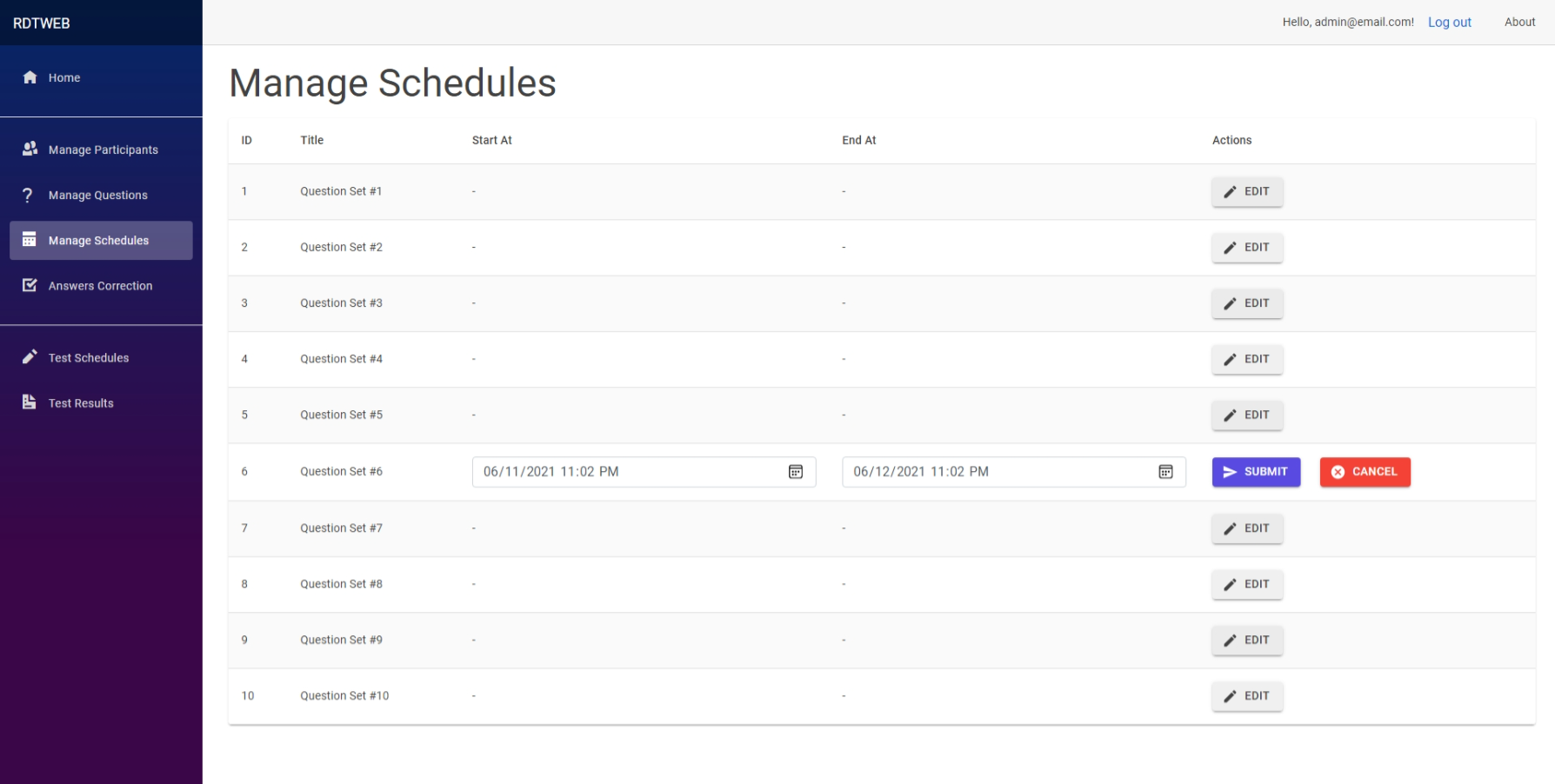


Figure 10. Manage schedules.

Admin can edit start and end time for each question sets listed in this page.

### Answer Corrections

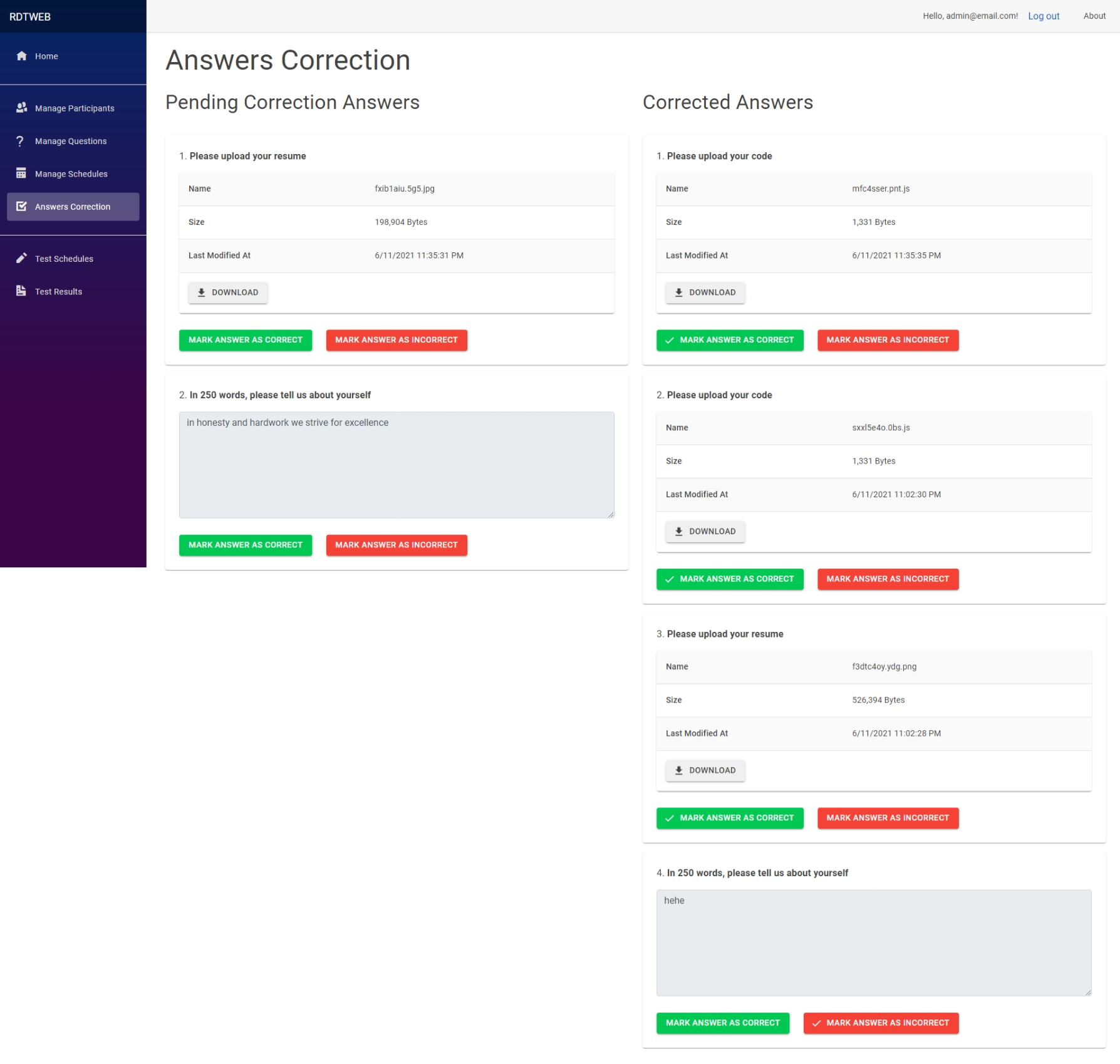


Figure 11. Answer corrections.

Admin can mark an answer to be either correct or incorrect. Questions shown here are only of type “Essay” and “Submit File”.

## Test Schedules

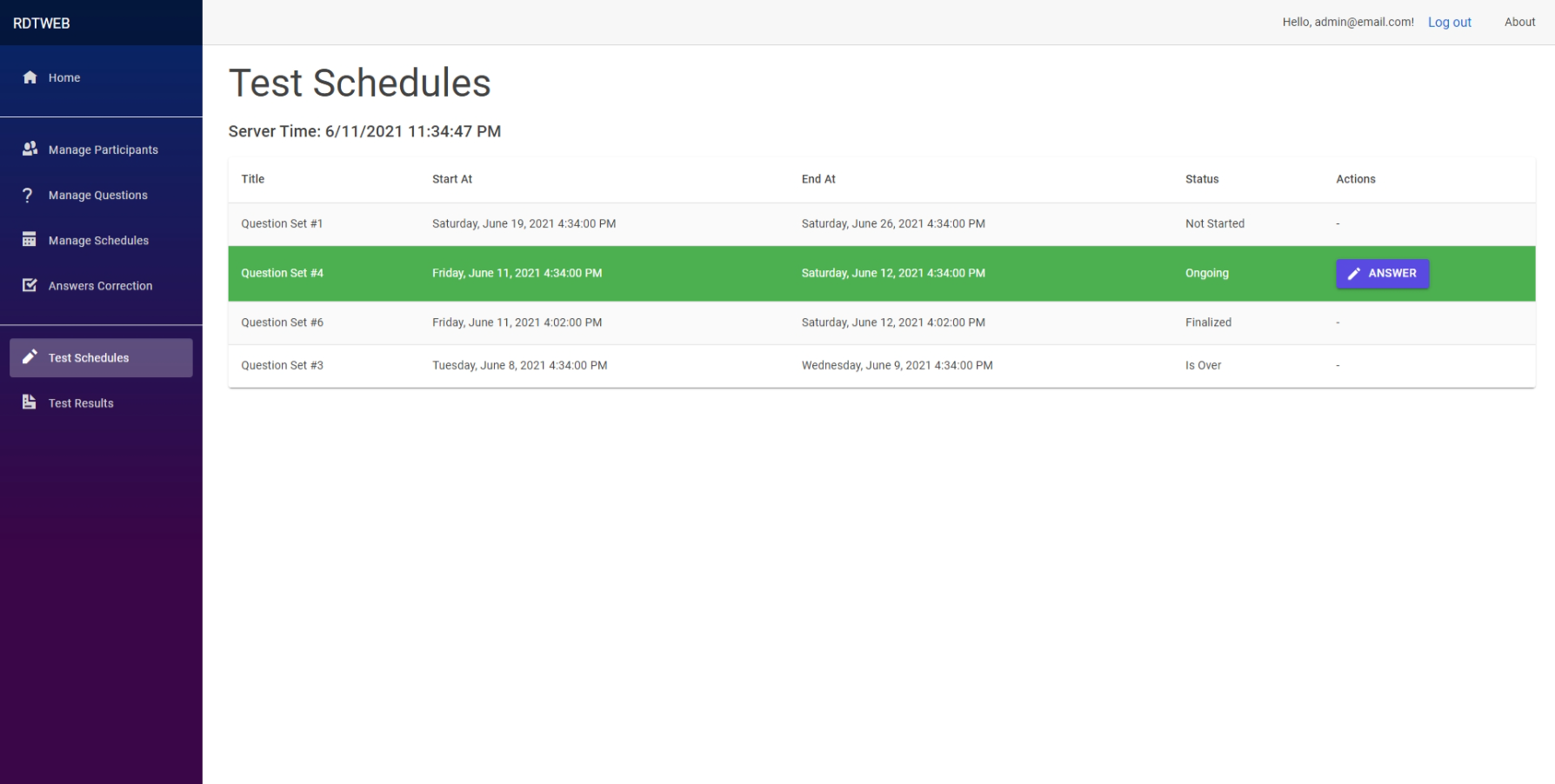


Figure 12. Test schedules.

Users can view all question sets that are either one of “Not Started”, “Ongoing”, “Is Over”, or “Finalized”. Users can attempt to answer an “Ongoing” question set and be taken to “Test Answer” page. Below explains about the test statuses:

* “Finalized” is when the user has finalized their answers in the test.
* “Not started” is when the start date is later than now.
* “Ongoing” is when now is later than start date and now is former than end date.
* “Is Over” is when now is later then end date.

## Test Answer

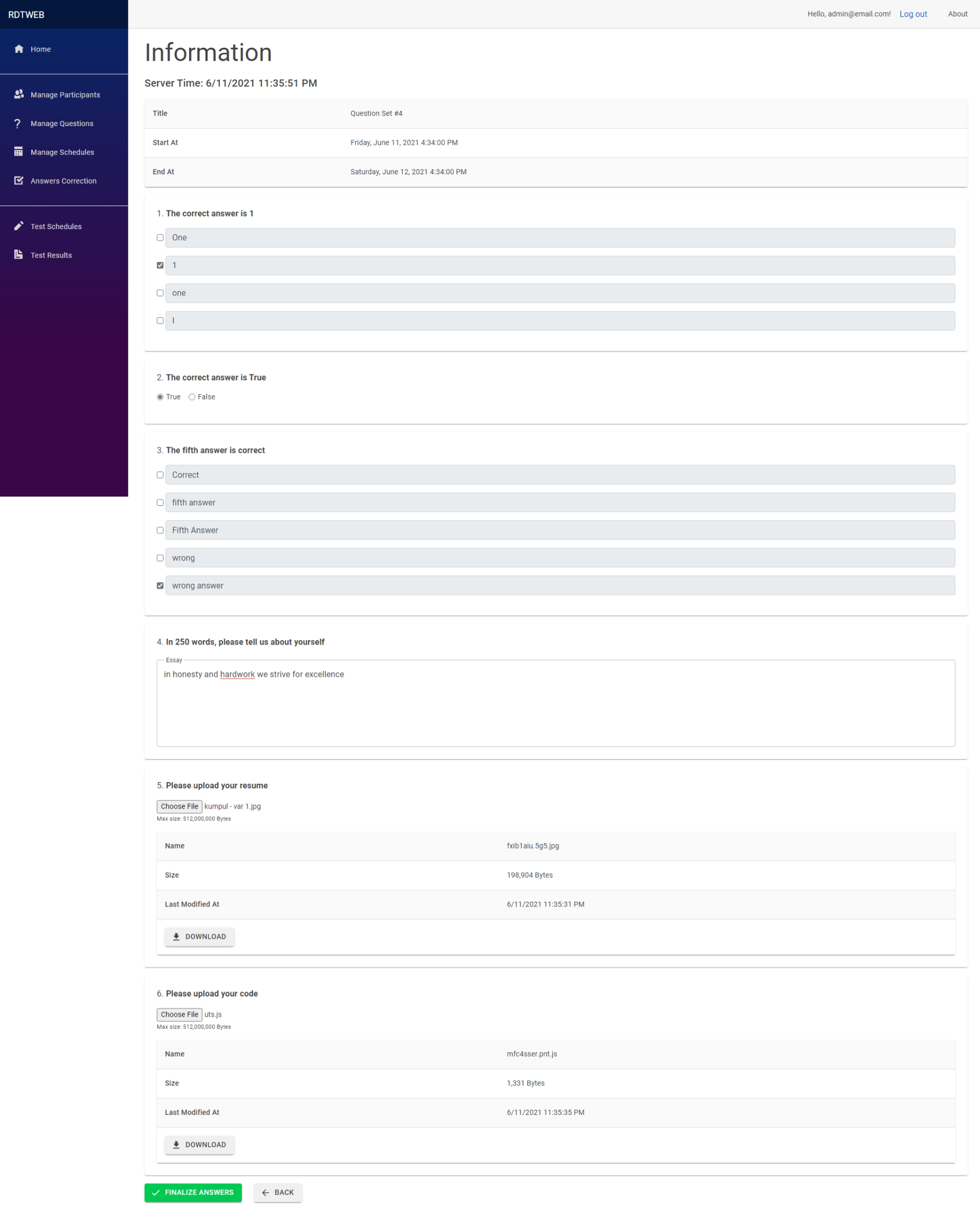


Figure 13. Test answer.

Users can answer a question set in this page. If a user has finalized their answer for the selected question set, then the user cannot update their answers anymore.

## Test Result

### Index

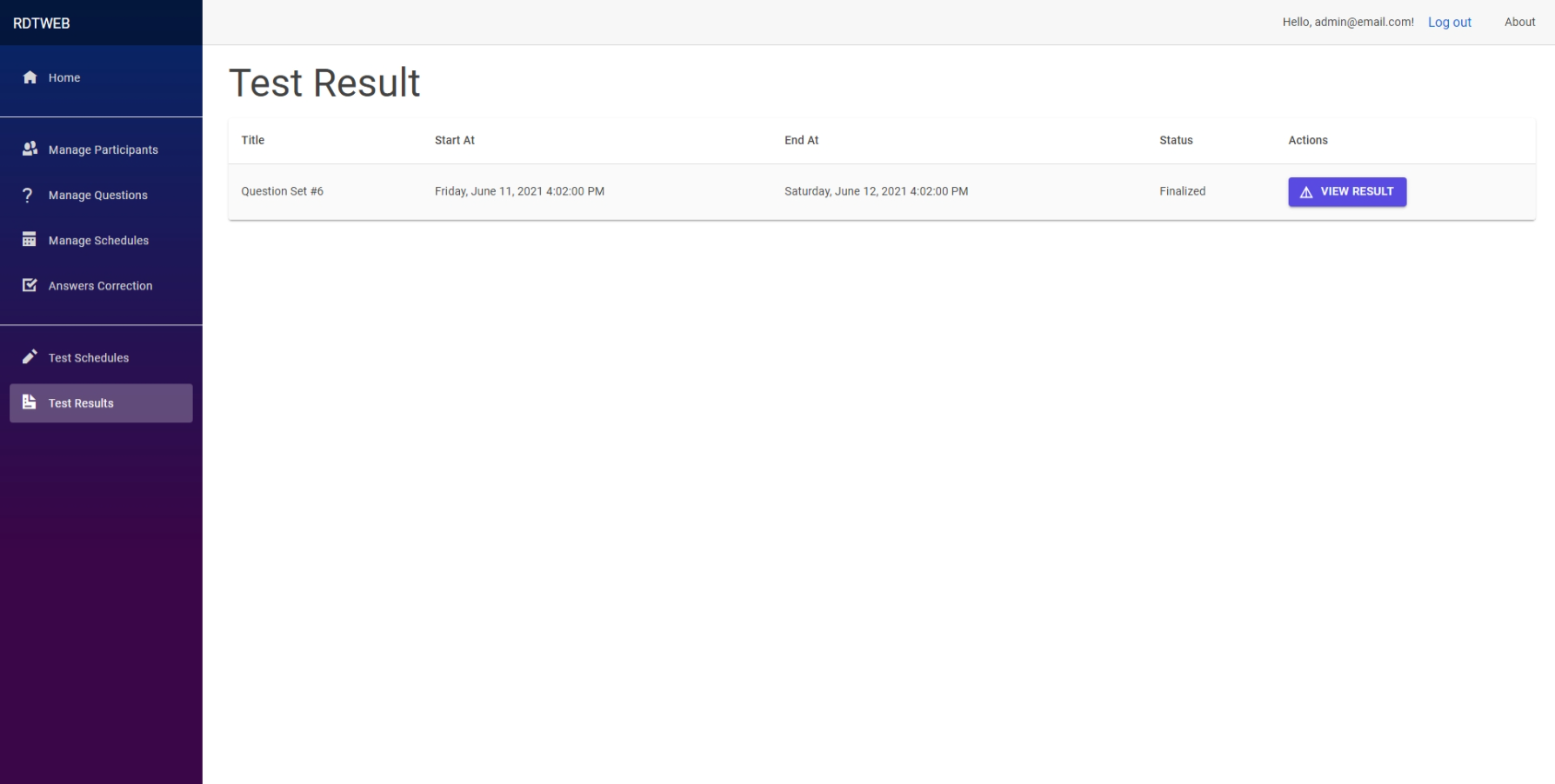


Figure 14. Test result index.

User can view question sets that they already answered and finalized. If the user has not finalized their answers, they will not be shown in this page.

### View

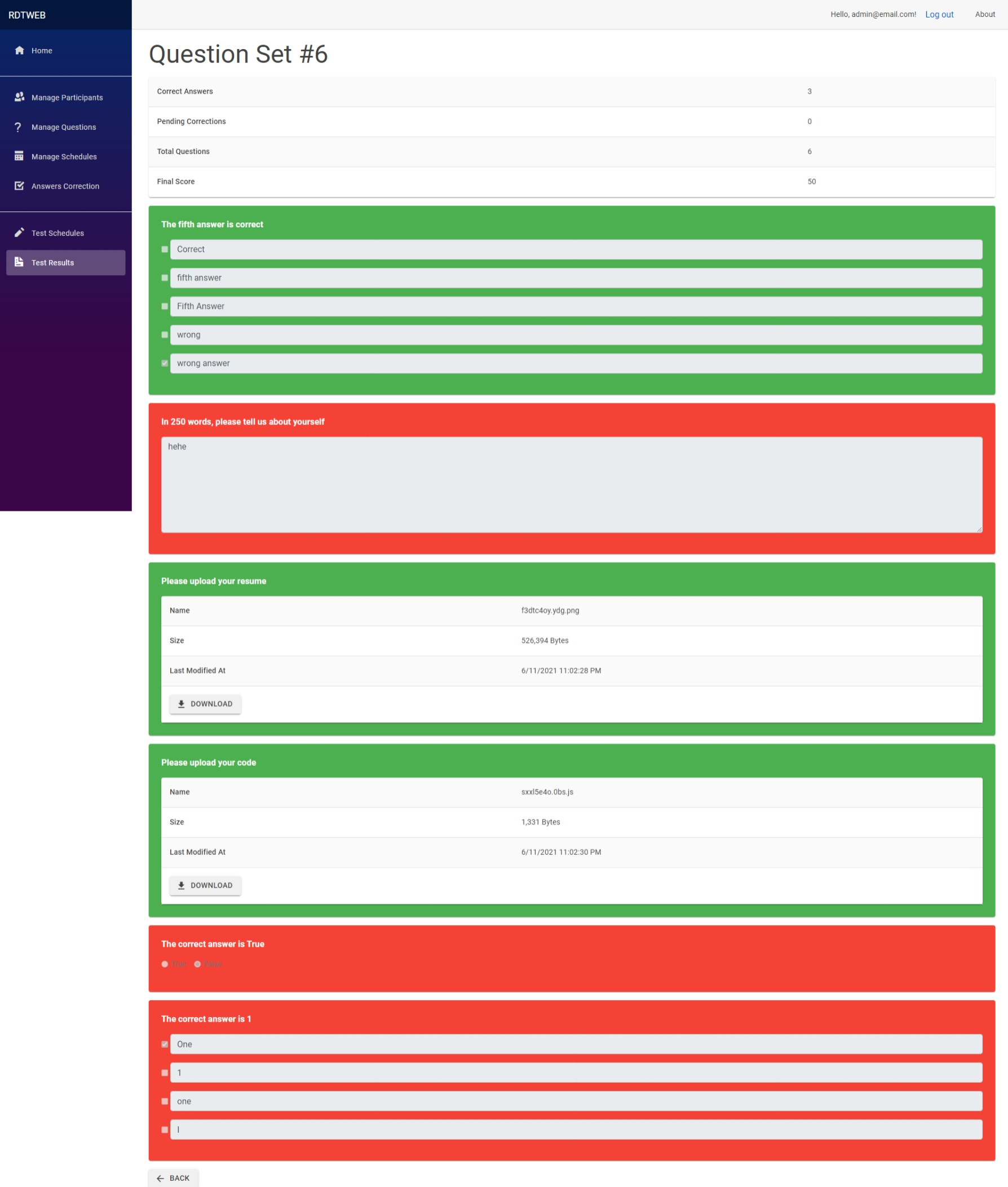


Figure 15. Test result view.

User can view their answers in a question set and view their answer status, that is either correct, incorrect, or pending correction. Grading information is also presented in this page.

# Performance Consideration

## Split Query

Entity Framework Core’s split query[[1]](#footnote-1) feature is used to prevent “cartesian explosion[[2]](#footnote-2)” problem when querying entities with one-to-many relationship. Hence, the solution is to split the query. The application enables split query globally to fix this problem in “Startup.cs”.

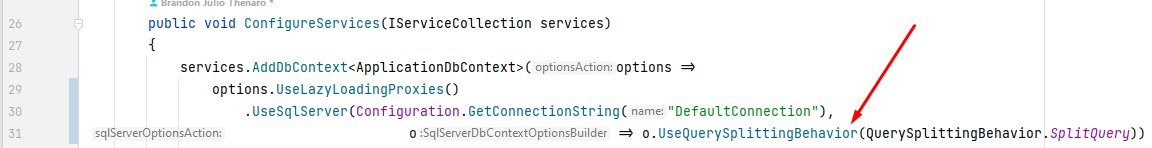


Figure 16. Split query configuration in "Startup.cs".

## Eager Loading

Relationships are eagerly loaded and explicitly specified to only load data which are required. Lazy loading is also used but sparingly to prevent N+1 problem, e.g., while looping[[3]](#footnote-3).

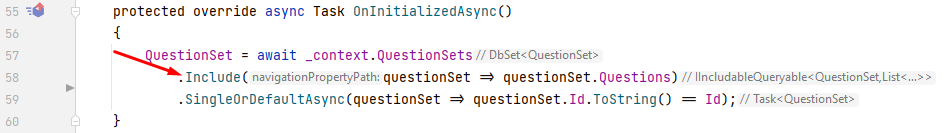


Figure 17. An example of explicit eager loading in” ManageQuestions/Index.razor”.

## Virtualize Component

Virtualize component is built-in Blazor component to improve UI performance while rendering list by limiting UI rendering only to parts that are currently visible[[4]](#footnote-4).



Figure 18. <Virtualize /> example in "ManageParticipants.razor".

## Asynchronous Programming

Asynchronous programming is applied while coding this application to prevent blocking the UI thread[[5]](#footnote-5) so that the UI is responsive to user’s input and feels snappy.

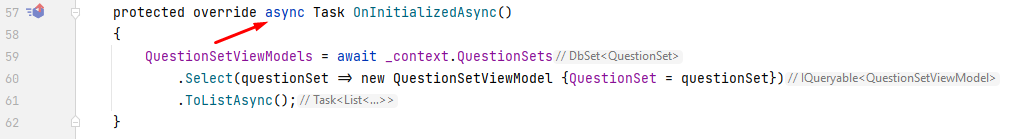


Figure 19. Asynchronous programming example in "ManageQuestions/Index.razor".

# Notes

* The .NET version used while developing the app is .NET 5.0.301.
* Admin account is [admin@email.com](mailto:admin@email.com) with password “d3F@uLtP@55w0Rd”.
* Please relog after seeding to avoid errors while answering questions.
* Every created user has default password of “d3F@uLtP@55w0Rd”.
* Default password can be changed in “appsettings.json”.
* Run seeding on every launch can be configured in “appsetting.json”.
* Used DBMS is SQL Server with database named “rdt” and uses Windows Authentication.
* Repository for the application can be found at <https://github.com/brandon-julio-t/RDTWEB>.

# ERD

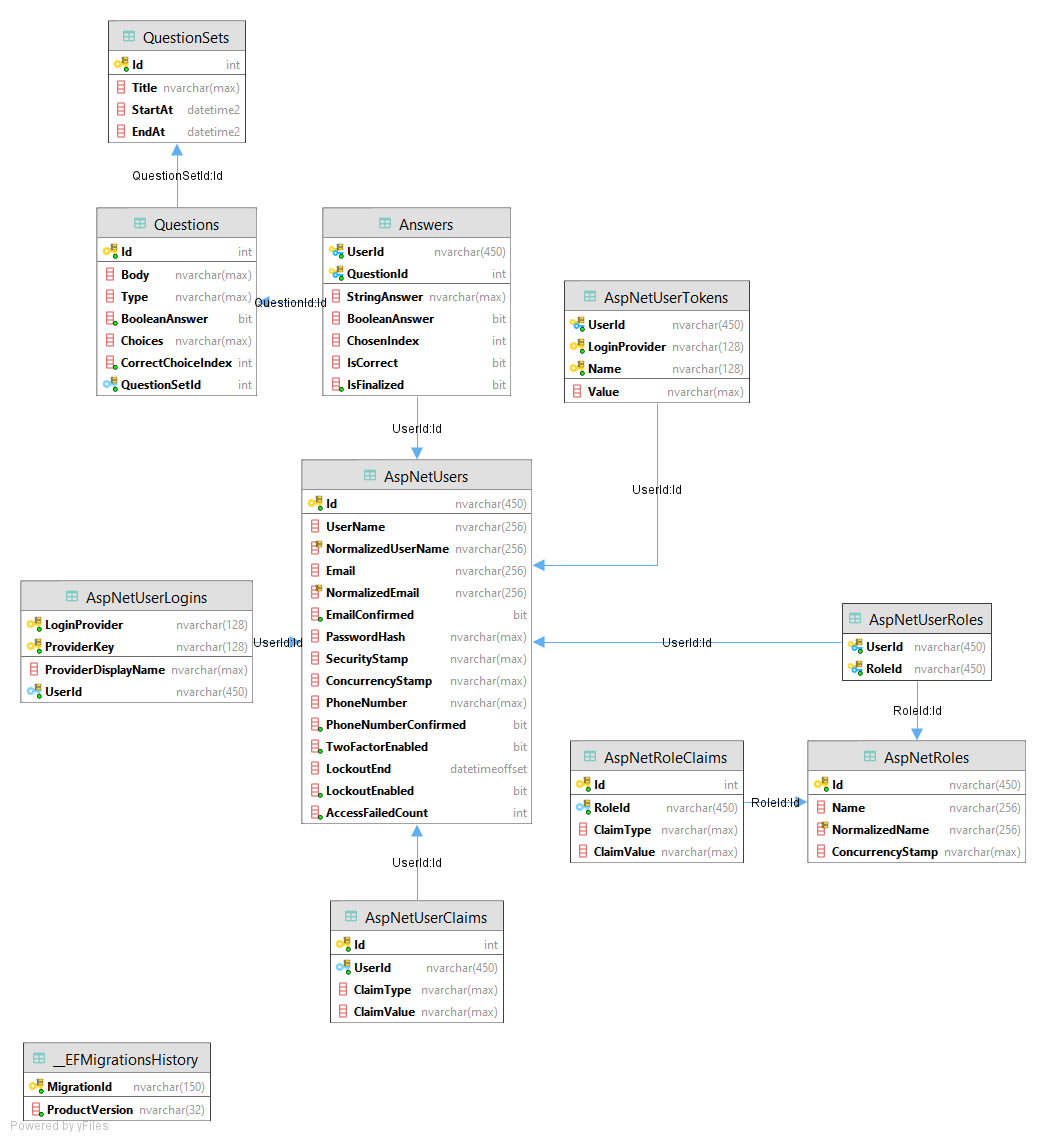


Figure 20. ERD

# References

[ASP.NET Web Apps | HTML, CSS, JavaScript, and C# (microsoft.com)](https://dotnet.microsoft.com/apps/aspnet/web-apps)

[Choose between ASP.NET 4.x and ASP.NET Core | Microsoft Docs](https://docs.microsoft.com/en-us/aspnet/core/fundamentals/choose-aspnet-framework?view=aspnetcore-5.0#framework-selection)

[Introduction to Identity on ASP.NET Core | Microsoft Docs](https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-5.0&tabs=visual-studio)

[Blazor | Build client web apps with C# | .NET (microsoft.com)](https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor)

[ASP.NET Core Blazor hosting models | Microsoft Docs](https://docs.microsoft.com/en-us/aspnet/core/blazor/hosting-models?view=aspnetcore-5.0)

[Overview of Entity Framework Core - EF Core | Microsoft Docs](https://docs.microsoft.com/en-us/ef/core/)

[Introduction · Bootstrap (getbootstrap.com)](https://getbootstrap.com/docs/4.3/getting-started/introduction/)

# Table of Contents

[Foreword 2](#_Toc74259532)

[Technologies 3](#_Toc74259533)

[ASP.NET Core 5.0 3](#_Toc74259534)

[ASP.NET Core Identity 3](#_Toc74259535)

[Blazor (Server) 3](#_Toc74259536)

[Entity Framework Core + SQL Server 3](#_Toc74259537)

[Bootstrap 4.3.1 3](#_Toc74259538)

[Pages 4](#_Toc74259539)

[Admin 4](#_Toc74259540)

[Manage Participants 6](#_Toc74259541)

[Manage Questions 6](#_Toc74259542)

[Manage Schedules 10](#_Toc74259543)

[Answer Corrections 10](#_Toc74259544)

[Test Schedules 11](#_Toc74259545)

[Test Answer 12](#_Toc74259546)

[Test Result 13](#_Toc74259547)

[Index 13](#_Toc74259548)

[View 14](#_Toc74259549)

[Performance Consideration 15](#_Toc74259550)

[Split Query 15](#_Toc74259551)

[Eager Loading 15](#_Toc74259552)

[Virtualize Component 15](#_Toc74259553)

[Asynchronous Programming 16](#_Toc74259554)

[Notes 17](#_Toc74259555)

[ERD 18](#_Toc74259556)

[References 19](#_Toc74259557)

[Table of Contents 20](#_Toc74259558)

1. [Single vs. Split Queries - EF Core | Microsoft Docs](https://docs.microsoft.com/en-us/ef/core/querying/single-split-queries) [↑](#footnote-ref-1)
2. [The performance issue "Cartesian Explosion" made its comeback in Entity Framework Core 3 - Thinktecture](https://www.thinktecture.com/en/entity-framework-core/cartesian-explosion-problem-in-3-1/) [↑](#footnote-ref-2)
3. [Efficient Querying - EF Core | Microsoft Docs](https://docs.microsoft.com/en-us/ef/core/performance/efficient-querying#beware-of-lazy-loading) [↑](#footnote-ref-3)
4. [ASP.NET Core Blazor component virtualization | Microsoft Docs](https://docs.microsoft.com/en-us/aspnet/core/blazor/components/virtualization?view=aspnetcore-5.0) [↑](#footnote-ref-4)
5. [Asynchronous Programming - EF Core | Microsoft Docs](https://docs.microsoft.com/en-us/ef/core/miscellaneous/async) [↑](#footnote-ref-5)