Tests to be done

Devices: IPhone 4s(480x320), IPhone 7(1920x1080), Macbook, PC 2core 3GHz (1920x1080), Asus i7 Intel(R) Core 2.6 GHz 8 GB RAM, Android (1920x1080) , HP Notebook Intel(R) Core i3-3110M 2.4 GHz (1366x768)

Test: Interface behavior on different platforms
Purpose: Responsiveness, Usability, Portability
Type: Manual
Expected result: Buttons and important information should be easy to access and read from every platform. No text out of bounds. On mobile devices should not be possible to download a form to be edited offline. An user can login, signup, view form templates, edit form and see responses from every platform. On PC, an user can download and edit offline a form. Users can fill a form(respond to a form) on every platform.
Possible problems: Blocks with information are not displayed properly on every platform. User can download form on mobile. Objects are not scaled appropriately to screen dimensions. Buttons hard to reach on mobile devices. Application is displayed very differently in different browsers. Hard to fill text fields on mobile devices. Design decisions that make the application hard to use.

Test: Login & signup
Purpose: Usability, Data consistency, Security, Portability
Type: Manual
Expected result: Username and password field should not allow illegal characters (https://support.google.com/a/answer/33386?hl=en). User must receive error messages when he is doing something illegal. When a new user is created, in the database must appear the informations introduced. The data entered in response must respect the constraints imposed on every field. The database stores the responses as filled by user, no data is altered. A user can change his password.
Possible problems: Username and password fields can accept all kinds of characters. When we try to create an user it is not inserted in the database. The created user doesn't correspond with the data in the database. Not all the error messages appear. A user cannot login with the username and password he just created.

Test: Fill form(respond)
Purpose: Usability, Data consistency
Type: Manual
Expected result: The filled form must be stored in the database and the data entered must correspond with the data in the database. User must receive error messages when he tries to introduce data that doesn't correspond with the constraints of the field.
Possible problems: User introduces more characters than the database stores. User try to introduce illegal data. User doesn't receive error messages. The submit button doesn't work. The data isn't stored in the database. What is stored in the database doesn't correspond with the filled form.

Test: Edit form online
Purpose: Data consistency, Responsiveness, Usability
Type: Manual
Expected result: User can move around a field. User can select the field data type. User can impose constraints on the field. User can add a question, a checkbox, a radio box, a spinner, a date.
Possible problems: User cannot move fields. User cannot delete/edit current fields. User cannot add constraints. The form is not saved properly. Additional unwanted information appear in the from.

Test: Edit form offline
Purpose: Data consistency, Responsiveness, Usability
Type: Manual
Expected result: All the new added fields should be saved among the old unedited fields
Possible problems: Editing a field could make odd repercussions among the other fields

Test: Multiple users respond to a form
Purpose: Scalability, Data consistency
Type: Automated
Expected result: Little to no influence on server performance. All the data received in order. All the data stored in the database.
Possible problems: Server cannot process that amount of data. The data received by server is altered. No all data end up in the database.

Test: Multiple users edit a form
Purpose: Scalability, Data consistency
Type: Automated
Expected result: Little to no influence on server performance. Forms are saved in the database. All the data fits user specifications.
Possible problems: Users cannot connect to server to edit forms. Server is very slow. No all the forms are saved. Missing data in the forms.

Test: Share link
Purpose: Data consistency, Usability, Security
Type: Manual
Expected result: User presses the share button and receives a link for other people to fill his form. User can share his form via social networks. User can share form to specific persons.
Possible problems: The link doesn’t work. Link directs the user to wrong form. Share via social networks doesn’t work. Share to specific persons can be accessed by everyone.

Test: Create, modify, download, delete form
Purpose: Data consistency, Usability
Type: Automated

Expected result: User can create a new form from an existing template or he can create a blank form. User can modify an existing form. User can download form. User can delete form.
Possible problems: Create button, edit button, download button, delete button doesn't work. The buttons are not easy to reach from mobile devices. User creates a new form but it is not stored in the database. Downloaded file doesn't correspond with the data in the database. User cannot modify an existing form. User modifies a form and the changes are not stored in the database. User deletes form and the form is still stored in the database.

Test: Input/Output of every method
Purpose: Unit testing, Data consistency
Type: Automated
Expected response: Every method should have a specific output for an input. The method should provide an output accordingly to the problem domain or throw an error if the input is not from the problem domain.
Possible problems: The output doesn't correspond with the expected output. The method has an output for an invalid input.

Test: Code review
Purpose: Unit testing, Quality assurance
Type: Manual
Expected response: Every method and attribute should respect the naming conventions, should be well commented
Possible problems: Ambiguous functions or variables, spaghetti code

Test: Interactions between objects and classes
Purpose: Usability
Type: Automated
Expected response: There shouldn't be unexpected results in the interactions between objects and classes. Every class should be in relation with other classes but it should be a clear difference between all classes
Possible problems: Redundant classes

Test: Time consuming, redundant operations, loading time
Purpose: Usability
Type: Automated
Expected response: The application should load quickly without having the user realise he is waiting too long
Possible problems: The application is moving too slow between the views and performing certain operations

Test: Graphical user interface
Purpose: Appeal
Type: Manual

Expected response: The views should be simple, not too overloaded with easy interaction features between them
Possible problems: The application is designed to be too agglomerate, making it difficult to navigate between the views