**SERVER SIDE MODULE**

# **Contents**

1. MODULE\_SERVER\_SIDE.docx
2. MODULE\_SERVER\_SIDE.pdf
3. MODULE\_SERVER\_SIDE\_MEDIA.zip

# **Introduction**

Your company “Formify Inc.” points you as a Web Developer to build a website which is possible for users to create form dynamically according to the question types such as short answer, paragraph, date input, multiple choice, dropdown and checkboxes. Users who created the form can share the form link to the user to submit a response of the form and also see all of the responses.

# **Description of Projects**

There are 2 phases in this module. In the first phase, you will make the RESTFul API using Laravel Frameworks according to the provided documentation. In the second phase, you should build a frontend with the Interactive Maps using one of the provided Javascript Frameworks (React/Vue/Angular) and the data must come from the created RESTFul API.

You can find the provided media files to support your work:

* Postman collection and environment
* Api tests (to testing your endpoints automatically)
* Template GUI (to build frontend UI)
* Formify.sql (a database with structures and dummy data)

# **Phase 1 : RESTful API**

In this phase, you should build a RESTful API using the Laravel framework according to the documentation below.

Ensure users can login using credentials below:

| **Name** | **Email** | **Password** |
| --- | --- | --- |
| User 1 | user1@webtech.id | password1 |
| User 2 | user2@webtech.id | password2 |
| User 3 | user3@worldskills.org | password3 |

## **Authentication**

You should create Login and Logout endpoints. The **accessToken must be generated by sanctum** and will be placed in the request headers **Authorization Bearer**.

### **Login**

| Endpoint | : | [DOMAIN]/api/v1/auth/login |
| --- | --- | --- |
| Description | : | For user to log in the system |
| Method | : | POST |
| Request | : | Body (JSON):  {  "email": "user1@webtech.id",  "password": "password1"  } |
| Validation | : | email:   * required * valid email address   password:   * required * min 5 chars |
| Response | : | **If success:** Headers:   * Response status: 200   Body (JSON):  {  "message": "Login success",  "user": {  "name": "User 1",  "email": "user1@webtech.id",  "accessToken": "1|QnHFgF0C7ml2LkbDG5QDn34qvJoYGdpxoXy63Poi"  }  } **If invalid field:** Headers:   * Response status: 422   Body (JSON):  {  "message": "Invalid field",  "errors": {  "email": [  "The email must be a valid email address."  ],  "password": [  "The password field is required."  ]  }  } **If failed:** Headers:   * Response status: 401   Body (JSON):  {  "message": "Email or password incorrect"  } |

### **Logout**

| Endpoint | : | [DOMAIN]/api/v1/auth/logout |
| --- | --- | --- |
| Description | : | For user to log out the system |
| Method | : | POST |
| Request | : | Headers:   * Authorization: “Bearer <accessToken>” |
| Response | : | **If success:** Headers:   * Response status: 200   Body (JSON):  {  "message": "Logout success"  } **If invalid token:** Headers:   * Response status: 401   Body (JSON):  {  "message": "Unauthenticated."  } |

## **Form**

Users can manage (create, see all created forms, and see detail form) with questions to share to invited users based on email domain to submit.

### **Create a Form**

| Endpoint | : | [DOMAIN]/api/v1/forms |
| --- | --- | --- |
| Description | : | For user to create new form |
| Method | : | POST |
| Request | : | Headers:   * Authorization: “Bearer <accessToken>”   Body (JSON):  {  "name": "Stacks of Web Tech Members",  "slug": "member-stacks",  "allowed\_domains": [ "webtech.id" ],  "description": "To collect all of favorite stacks",  "limit\_one\_response": true  } |
| Validation | : | name:   * required   slug:   * required * unique * alphanumeric with special characters only dash “-” and dot “.” and without space   allowed\_domains:   * array |
| Response | : | **If success:** Headers:   * Response status: 200   Body (JSON):  {  "message": "Create form success",  "form": {  "name": "Stacks of Web Tech Members",  "slug": "member-stacks",  "description": "To collect all of favorite..",  "limit\_one\_response": true,  "creator\_id": 1,  "id": 3  }  } **If invalid field:** Headers:   * Response status: 422   Body (JSON):  {  "message": "Invalid field",  "errors": {  "name": [  "The name field is required."  ],  "slug": [  "The slug has already been taken."  ],  "allowed\_domains": [  "The allowed domains must be an array."  ]  }  } **If invalid token:** Headers:   * Response status: 401   Body (JSON)  {  "message": "Unauthenticated."  } |

### **Get all Forms**

| Endpoint | : | [DOMAIN]/api/v1/forms |
| --- | --- | --- |
| Description | : | For users to get all of their created forms |
| Method | : | GET |
| Request | : | Headers:   * Authorization: “Bearer <accessToken>” |
| Response | : | **If success:** Headers:   * Response status: 200   Body (JSON):  {  "message": "Get all forms success",  "forms": [  {  "id": 1,  "name": "Biodata - Web Tech Members",  "slug": "biodata",  "description": "To save web tech membe..",  "limit\_one\_response": 1,  "creator\_id": 1  },  {  "id": 2,  "name": "HTML and CSS Skills - Quiz",  "slug": "htmlcss-quiz",  "description": "Fundamental web tests",  "limit\_one\_response": 1,  "creator\_id": 1  },  {  "id": 3,  "name": "Stacks of Web Tech Members",  "slug": "member-stacks",  "description": "To collect all of favorit..",  "limit\_one\_response": 1,  "creator\_id": 1  }  ]  } **If invalid token:** Headers:   * Response status: 401   Body (JSON):  {  "message": "Unauthenticated."  } |

### **Detail Form**

| Endpoint | : | [DOMAIN]/api/v1/forms/<form\_slug> |
| --- | --- | --- |
| Description | : | For invited users to get detail form |
| Method | : | GET |
| Request | : | - |
| Response | : | **If success:** Headers:   * Response status: 200   Body (JSON):  {  "message": "Get form success",  "form": {  "id": 1,  "name": "Biodata - Web Tech Members",  "slug": "biodata",  "description": "To save web tech members data",  "limit\_one\_response": 1,  "creator\_id": 1,  "allowed\_domains": [  "webtech.id"  ],  "questions": [  {  "id": 1,  "form\_id": 1,  "name": "Name",  "choice\_type": "short answer",  "choices": null,  "is\_required": 1  },  {  "id": 2,  "form\_id": 1,  "name": "Address",  "choice\_type": "paragraph",  "choices": null,  "is\_required": 0  },  {  "id": 3,  "form\_id": 1,  "name": "Born Date",  "choice\_type": "date",  "choices": null,  "is\_required": 1  },  {  "id": 4,  "form\_id": 1,  "name": "Sex",  "choice\_type": "multiple choice",  "choices": "Male,Female",  "is\_required": 1  }  ]  }  } **If form slug invalid:** Headers:   * Response status: 404   Body (JSON):  {  "message": "Form not found"  } **If user email domain not allowed to submit form:** Headers:   * Response status: 403   Body (JSON):  {  "message": "Forbidden access"  } **If invalid token:** Headers:   * Response status: 401   Body (JSON):  {  "message": "Unauthenticated."  } |

## 

## **Question**

Users can manage questions (add and remove) from one of their created forms. Choices field must be required if the user selects multiple choice, dropdown, or checkboxes. There are 6 choice types:

1. Short answer (TextField)
2. Paragraph (TextArea)
3. Date (Input Date)
4. Multiple Choice (Radio) : with choices
5. Dropdown (Select) : with choices
6. Checkboxes (Checkboxes) : with choices

### **Add a Question**

| Endpoint | : | [DOMAIN]/api/v1/forms/<form\_slug>/questions |
| --- | --- | --- |
| Description | : | For user to add a question of a form |
| Method | : | POST |
| Request | : | Headers:   * Authorization: “Bearer <accessToken>”   Body (JSON):  {  "name": "Most Favorite JS Framework",  "choice\_type": "multiple choice",  "choices": [  "React JS",  "Vue JS",  "Angular JS",  "Svelte"  ],  "is\_required": true  } |
| Validation | : | name:   * required   choice\_type:   * required * only: “short answer”, “paragraph”, “date”, “multiple choice”, “dropdown”, or “checkboxes”   choices   * required if selected choice type is “multiple choice”, “dropdown”, or “checkboxes” |
| Response | : | **If success:** Headers:   * Response status: 200   Body (JSON):  {  "message": "Add question success",  "question": {  "name": "Most Favorite JS Framework",  "choice\_type": "multiple choice",  "is\_required": true,  "choices": "React JS,Vue JS,Angular JS,Svelte",  "form\_id": 3,  "id": 10  }  } **If invalid field:** Headers:   * Response status: 422   Body (JSON):  {  "message": "Invalid field",  "errors": {  "name": [  "The name field is required."  ],  "choice\_type": [  "The choice type field is required."  ],  "choices": [  "The choices must be an array.",  "The choices must be a string."  ]  }  } **If invalid form slug:** Headers:   * Response status: 404   Body (JSON):  {  "message": "Form not found"  } **If try access another user form:** Headers:   * Response status: 403   Body (JSON):  {  "message": "Forbidden access"  } **If invalid token:** Headers:   * Response status: 401   Body (JSON):  {  "message": "Unauthenticated."  } |

### **Remove a Question**

| Endpoint | : | [DOMAIN]/api/v1/forms/<form\_slug>/questions/<question\_id> |
| --- | --- | --- |
| Description | : | For user to remove a question of a form |
| Method | : | DELETE |
| Request | : | Headers:   * Authorization: “Bearer <accessToken>” |
| Response | : | **If success:** Headers:   * Response status: 200   Body (JSON):  {  "message": "Remove question success"  } **If invalid form slug:** Headers:   * Response status: 404   Body (JSON):  {  "message": "Form not found"  } **If invalid question id:** Headers:   * Response status: 404   Body (JSON):  {  "message": "Question not found"  } **If try access another user form:** Headers:   * Response status: 403   Body (JSON):  {  "message": "Forbidden access"  } **If invalid token:** Headers:   * Response status: 401   Body (JSON):  {  "message": "Unauthenticated."  } |

## **Response**

If the creator form sets the form with limit 1 response, then the invited user can’t submit a response twice. Submit Response only can be accessed by users with an allowed domain of the user email. If the creator form does not fill allowed domains, it means that the form can be accessed by anyone/public.

**For example:**

An user with email “[user3@worldskills.org](mailto:user1@worldskills.org)” can’t submit a response to the form with allowed domains webtech.id and inaskills.id, because worldskills.org can't be allowed to access the form.

### **Submit Response**

| Endpoint | : | [DOMAIN]/api/v1/forms/<form\_slug>/responses |
| --- | --- | --- |
| Description | : | For invited user to send response of a form |
| Method | : | POST |
| Request | : | Headers:   * Authorization: “Bearer <accessToken>”   Body (JSON):  {  "answers": [  {  "question\_id": 1,  "value": "Ica Amalia"  },  {  "question\_id": 2,  "value": "Bandung"  },  {  "question\_id": 3,  "value": "2006-08-01"  },  {  "question\_id": 4,  "value": "Female"  }  ]  } |
| Validation | : | answers   * array   [question value]   * required if sets true |
| Response | : | **If success:** Headers:   * Response status: 200   Body (JSON):  {  "message": "Submit response success"  } **If invalid field:** Headers:   * Response status: 422   Body (JSON):  {  "message": "Invalid field",  "errors": {  "answers": [  "The answers field is required."  ]  }  } **If user email domain not allowed to submit form:** Headers:   * Response status: 403   Body (JSON):  {  "message": "Forbidden access"  } **If limited for 1 response then user submit twice:** Headers:   * Response status: 422   Body (JSON):  {  "message": "You can not submit form twice"  } **If invalid token** Headers:   * Response status: 401   Body (JSON):  {  "message": "Unauthenticated."  } |

### **Get all Responses**

| Endpoint | : | [DOMAIN]/api/v1/forms/<form\_slug>/responses |
| --- | --- | --- |
| Description | : | For the creator see all responses |
| Method | : | GET |
| Request | : | Headers:   * Authorization: “Bearer <accessToken>” |
| Response | : | If success: Headers:   * Response status: 200   Body (JSON):  {  "message": "Get responses success",  "responses": [  {  "date": "2022-10-24 01:47:14",  "user": {  "id": 1,  "name": "User 1",  "email": "user1@webtech.id",  "email\_verified\_at": null  },  "answers": {  "Name": "Budi Setiawan",  "Address": "Jakarta, Indonesia",  "Born Date": "2004-05-05",  "Sex": "Male"  }  },  {  "date": "2022-10-24 02:03:27",  "user": {  "id": 2,  "name": "User 2",  "email": "user2@webtech.id",  "email\_verified\_at": null  },  "answers": {  "Name": "Ica Amalia",  "Address": "Bandung",  "Born Date": "2006-08-01",  "Sex": "Female"  }  }  ]  } **If form slug invalid:** Headers:   * Response status: 404   Body (JSON):  {  "message": "Form not found"  } **If try access another user form:** Headers:   * Response status: 403   Body (JSON)  {  "message": "Forbidden access"  } **If invalid token** Headers   * Response status: 401   Body (JSON)  {  "message": "Unauthenticated."  } |

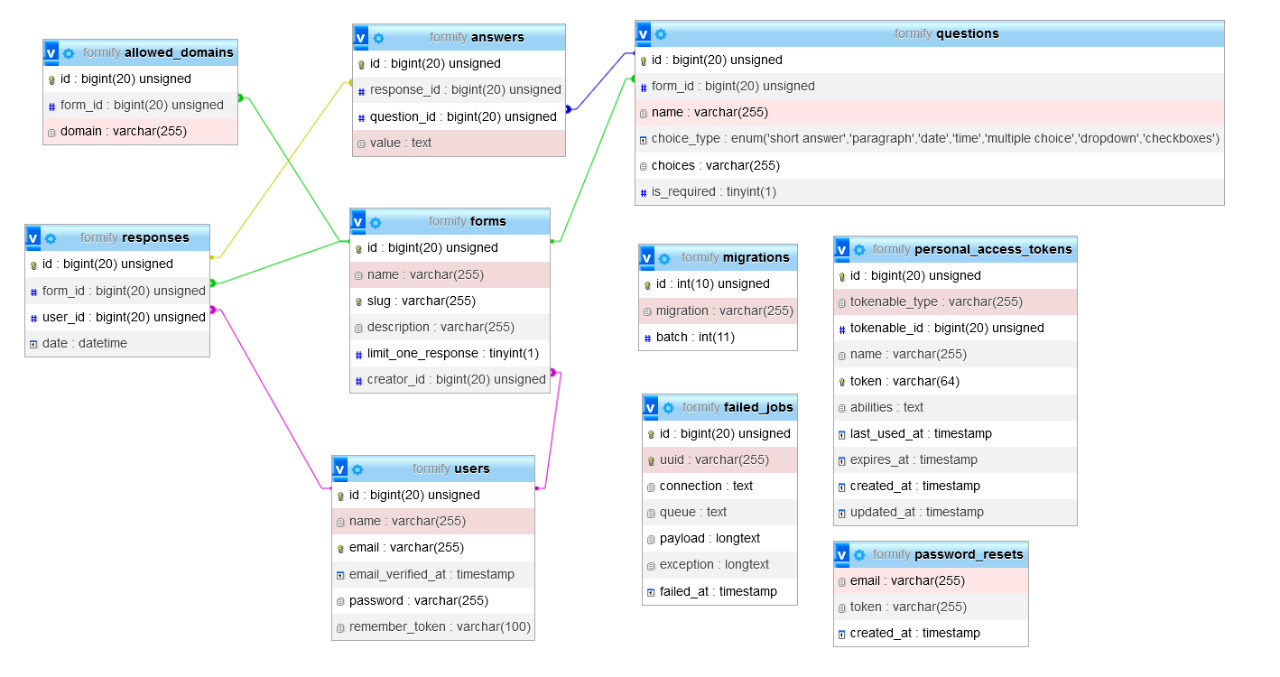
# 

# **Phase 2 : Front-end Development**

In this part, you should build a front-end application using one of the provided frameworks (vue js or react js). You can use the provided template gui on the media files to build the front-end ui.

| **Pages / Features** | **Description** |
| --- | --- |
| Login and Logout | * User can login using the correct credentials (username and password) * Alert errors should be displayed when login failed * Users can log out by clicking the logout button on the navbar. |
| Home | * Users can see their list of created forms. |
| Create Form | * Form inputs displayed correctly (name, slug, description, allowed domains and limit to 1 response switch). Name and slug fields are required. * Users can create a form by filling in all of the required fields. * Alert errors should be displayed when failed. |
| Detail Form | **General:**   * Form detail displayed correctly * Form link displayed correctly * User can copy form link   **Questions:**   * Questions displayed correctly * Question inputs should be disabled when has created * Users can add questions by filling in a question form and clicking the save button. * Choices input should be displayed when the user selects multiple choice, dropdown, or checkboxes. * Alert errors should be displayed when created failed. * Users can remove questions by clicking the remove button.   **Responses:**   * Table displayed correctly according to questions and responses. * Total responses count is displayed correctly. |
| Submit Form | * Form detail displayed correctly (name, description, and user email). * Questions with the form displayed correctly. * Users can submit by filling in all of the required forms and clicking the submit button. * Submit button should be disabled if there is a required form that hasn’t been filled. * Users can’t submit twice if the form is limited for 1 response. * Forbidden access page should be displayed when the user isn't invited to access that form. |

# **ERD**



# **Instructions**

* Save your REST Api in the directory called **“SERVER\_MODULE/backend”** and save it in the repository folder.
* Save your front-end app in the directory called **“SERVER\_MODULE/frontend”** and save it in the repository folder.
* When developing the Frontend, you can **use/consume the provided REST API**.
* You should **build the frontend app to production mode** and **save the production files in the “SERVER\_MODULE/frontend/build” folder** before you push it to the github repository.
* Make sure the **“/build”** folder **isn't written on the .gitgnore file**.
* You should commit and push your changes to the github repository at least every 30 minutes.

# **Marking Overview**

| **Phase** | **Parts** | **Measurement** | **Judgment** | **Total** |
| --- | --- | --- | --- | --- |
| REST API | Database | 1.00 | 0 | 1.00 |
| Authentication | 1.50 | 0 | 1.50 |
| Form | 3.50 | 0 | 3.50 |
| Question | 2.50 | 0 | 2.50 |
| Response | 5.00 | 0 | 5.00 |
| Framework | 0 | 1.50 | 1.50 |
| Front-end | Authentication | 1.00 | 0 | 1.00 |
| Manage Forms | 0.50 | 0 | 0.50 |
| Create Form | 1.25 | 0 | 1.25 |
| Detail Form | 6.00 | 0 | 6.00 |
| Submit Response | 3.75 | 0 | 3.75 |
| General | 0.50 | 0 | 0.50 |
| JS Quality | 0 | 2.00 | 2.00 |
| **Total** | | **26.50** | **3.50** | **30.00** |