# MODULE SERVER

# CONTENTS

**This module has the following files:**

* MODULE\_SERVER.doc
* MODULE\_SERVER\_MEDIA.zip

# INTRODUCTION

PT. XYZ decided to create simple anonymous voting system called **YukPilih**. This system will be used for asking some company decisions to their employees. Your task is to implement the backend with Laravel PHP Framework and frontend with JavaScript Framework (VueJS, AngularJS, or ReactJS). The detail description and tools that you can use will be described below.

## DESCRIPTION OF PROJECT AND TASKS

**API List:**

Use provided ERD to make your database. Create dummy users from provided “users.csv” to users and divisions table (password is hashed using bcrypt).

These are the list of web service endpoint that requested by the company:

1. **Authentication**
   1. **Login**

URL: [domain]/api/auth/login

Description: For client to generate and get login token using username and password. Username and password must be valid.

Method: POST

Request Parameter:

* Body:
  + username
  + password

Response:

* If login success:

Header: response status: 200

Body:

* + access\_token (authorization token generated by the system using JSON Web Token (JWT))
  + token\_type (always Bearer)
  + expires\_in (set your token to be expired in 24 hours)
* If login failed (username or password do not match or empty):

Header: response status: 401

Body: error: Unauthorized

* 1. **Logout**

URL: [domain]/api/auth/logout

Headers:

* Authorization: Bearer [access\_token]

Description: For server to make token invalid

Method: POST

Response:

* If logout success:

Header: response status: 200

Body:

* + message: successfully logged out
* If logout failed (token invalid):

Header: response status: 401

Body: message: Unauthorized

* 1. **Me**

URL: [domain]/api/auth/me

Headers:

* Authorization: Bearer [access\_token]

Description: Get logged in user data

Method: POST

Response:

* If success:

Header: response status: 200

Body:

* + User data including id, division\_id, username, role, created\_at, updated\_at
* If failed (token invalid):

Header: response status: 401

Body: message: Unauthorized

* 1. **Reset Password**

URL: [domain]/api/auth/reset\_password

Headers:

* Authorization: Bearer [access\_token]

Description: Reset user password. Invalidate current token if reset success.

Method: POST

Request Parameter:

* Body:
  + old\_password
  + new\_password

Response:

* If success:

Header: response status: 200

Body: message: reset success, user logged out

* If failed (old password did not match):

Header: response status: 422

Body: message: old password did not match

* If failed (token invalid):

Header: response status: 401

Body: message: Unauthorized

1. **Poll**
   1. **Create new poll**

URL: [domain]/api/poll

Headers:

* Authorization: Bearer [access\_token]

Description: Create new poll. Admin only.

Method: POST

Request Parameter:

* Body:
  + title: String, required.
  + description: String, required.
  + deadline: Datetime, required.
  + choices: array of string, choices must be unique. Validate at least two items given.

Response:

* If success:

Header: response status: 200

Body: JSON of the new poll data

Header: response status: 422

Body: message: The given data was invalid.

* If unauthorized user access it (only admin can access this endpoint):

Header: response status: 401

Body: message: Unauthorized

* 1. **Get all poll**

URL: [domain]/api/poll

Headers:

* Authorization: Bearer [access\_token]

Description: Get all poll data. Result only shown to admin or if user already voted for that poll or if that post already exceeds the deadline. Otherwise, result will be returned as null.

Method: GET

Response:

* If success:

Header: response status: 200

Body: array of JSON of the polls data, including id, title, description, deadline, created\_by (creator user id), created\_at, creator (creator username), result, and choices.

Notes: result will be an array consists of id (choice\_id), choice, and point. Check detailed formula in the end of this document to learn how the point calculated.

* If using invalid token:

Header: response status: 401

Body: message: Unauthorized

* 1. **Get a poll**

URL: [domain]/api/poll/[poll\_id]

Headers:

* Authorization: Bearer [access\_token]

Description: Get one poll data by given poll\_id. Result only shown to admin or if user already voted for that poll or if that post already exceeds the deadline. Otherwise, result will be returned as null.

Method: GET

Response:

* If success:

Header: response status: 200

Body: JSON of the polls data, including id, title, description, deadline, created\_by (creator user id), created\_at, creator (creator username), result, and choices.

Notes: result will be an array consists of id (choice\_id), choice, and point. Check detailed formula in the end of this document to learn how the point calculated.

* If using invalid token:

Header: response status: 401

Body: message: Unauthorized

* 1. **Vote**

URL: [domain]/api/poll/[poll\_id]/vote/[choice\_id]

Headers:

* Authorization: Bearer [access\_token]

Description: Logged user give a vote for given choice\_id to the given poll\_id. Admin cannot do a vote.

Method: POST

Response:

* If success:

Header: response status: 200

Body: message: voting success

* If user already voted for this poll:

Header: response status: 422

Body: message: already voted

* If poll already deadline:

Header: response status: 422

Body: message: voting deadline

* If user try to use invalid choice\_id or poll\_id:

Header: response status: 422

Body: message: invalid choice

* If unauthorized user access it (only user can access this endpoint):

Header: response status: 401

Body: message: Unauthorized

* 1. **Delete a poll**

URL: [domain]/api/poll/[poll\_id]

Headers:

* Authorization: Bearer [access\_token]

Description: Admin delete poll based on given poll\_id.

Method: DELETE

Response:

* If success:

Header: response status: 200

* If unauthorized user access it (only admin can access this endpoint):

Header: response status: 401

Body: message: Unauthorized

The complete **YukPilih** system should cover the following requirement:

|  |  |
| --- | --- |
| **Menu** | **Detail** |
| Login | * The system’s role can be an Admin or a User * The user (admin or user) can login (and logout) into the system on the start page of the application * After login, if admin or user still using default password, frontend will be redirected to change password page. Else, redirect to poll page. * After login, admin will have floating action button (FAB) to create a new poll * After login, user can view all schedule   Note: the system is not providing register menu, all user data registration is managed by administrator on the database directly |
| Change Password | * After login, if admin or user still using default password, frontend will be redirected to change password page * On this page, user will be asked to input old password, new password, and confirm new password. Validate new password and confirm new password must be match before hitting reset\_password backend endpoint. * After success resetting password, redirect to login page. |
| Poll | * Username and logout button will be displayed on the menu. Admin or user can use the button for logging out. * Admin will have floating action button (FAB) to create a new poll * Admin cannot do vote. Admin always seeing the poll results. * Admin can delete polls. * User can do vote only once per poll. * User only seeing poll results only if the poll already deadline or user already voted for that poll. * Polls sorted by created\_at descending. * Poll results will be shown as percentage with following formula:   Choice’s percentage = choice’s point / [total of all points] x 100%   * Point will be calculated with following standards:   + Regardless number of division members, a division will only have 1 point for each poll.   + All voting will be calculated internally on each division.   + After calculated internally, the winner from each division will be used for the whole poll.   + If there is more than one winner on a division, the division’s point will be split equally. |
| Create Poll Dialog | * If admin click the FAB button, this dialog will be shown. * Admin asked to fill title (string), description (string, can be multiline), deadline (use date time picker, default value is current time), and choices. * Choices have special behaviours:   + Initially, there is only one input field.   + After admin typing down on the input field, a new blank input field will be appended in the bottom of the form.   + Admin can delete input fields, but make sure admin always have at least one blank input field in the bottom of the form. * There is a cancel button to close this dialog without creating poll. |

## YukPilih Point Calculation Simulation

**Simulation 1:**

This poll having 2 choices (WFO or WFH).

* Two users from payment division already vote. All of them choosing WFO. WFO win on this division, one point for WFO.
* One user from procurement division choosing WFO. WFO win on this division, one point for WFO.
* Six users from IT division already vote. 1 choosing WFO, 5 choosing WFH. WFH win on this division, one point for WFH.
* Three users from finance division already vote. All of them choosing WFH. WFH win on this division, one point for WFH.

Result: WFO 2 points. WFH 2 points. 50% percentage for each option will be displayed on the frontend.

Graphical user interface, text, application

Description automatically generated

**Simulation 2:**

This poll having 3 choices (‘senin’, ‘rabu’, or ‘jumat’).

* All users from payment division already vote. All of them choosing ‘rabu. ‘rabu’ win on this division, one point for ‘rabu’.
* Three users from procurement division voted. 1 choosing ‘senin’, 1 choosing ‘rabu’, 1 choosing ‘jumat’. We have 3 winners on this division. 1/3 point will be given for each option.
* Five users from IT division voted. 2 choosing ‘senin’, 2 choosing ‘rabu’, 1 choosing ‘jumat’. We have 2 winners on this division. 1/2 point will be given for ‘senin’ and ‘rabu’.
* No user from finance division voted. No point given from this division yet.

Result: ‘senin’ 0.8333 points. ‘rabu’ 1.8333 points. ‘jumat’ 0.3333 points.

Percentage calculation:

‘senin’ = 0.8333 / (0.8333 + 1.8333 + 0.3333) \* 100% = 27.78%

‘rabu’ = 1.8333 / (0.8333 + 1.8333 + 0.3333) \* 100% = 61.11%

‘jumat’ = 0.3333 / (0.8333 + 1.8333 + 0.3333) \* 100% = 11.11%

Graphical user interface, text, application, email

Description automatically generated

## ERD

You can use and improve ERD below:

Diagram

Description automatically generated

## INSTRUCTIONS TO THE COMPETITOR

* Save your files in your root directory on the server called "**module\_server**"

## Create/generate a DB-diagram named “db-diagram.xxx” (xxx is the extension/type of the file eg. pdf or jpg) and put it into the directory mentioned above. Example: Diagram Description automatically generated

# For this module, you must use one of the three available frameworks provided. Applications developed without use of any of these frameworks will not be considered. You should take advantage of the framework as much as possible.