*LKS SMK*
*Tingkat Kota Tangerang*
*Tahun 2019*

LKS SMK
Tingkat Kota Tangerang
2019

**Soal**

BIDANG LOMBA

# *Web Design and Development*

SMK Bisa !
Indonesia Bisa !

# CMS

## Introduction

GLOBAL FASHION 2017 IS A BRAND NEW FASHION EVENT THAT IS TO BE HELD IN ABU DHABI. THE COMPANY RUNNING THIS EVENT NEEDS HELP IN DESIGNING A NEW AND VIBRANT WEBSITE IN ORDER TO MARKET THE EVENT AND SELL TICKETS.

## Description of project and tasks

THIS MODULE INVOLVES KNOWLEDGE ABOUT WEBSITE DESIGN, WEBSITE LAYOUT TECHNIQUES, CLIENT-SIDE SCRIPTING, AND SERVER-SIDE SCRIPTING, ALL COMBINED IN ONE CMS PROJECT. YOU WILL BE USING MOST POPULAR CMS, WORDPRESS. YOU WILL DEVELOP YOUR THEME AS A CHILD-THEME FOR THE BLANKSLATE WORDPRESS THEME, CALLED *BLANKSLATE_CHILD_WORLDFASHION*. FOR THE FULLY FUNCTIONAL WEBSITE YOU WILL NEED TO ADD SOME OF THE PROVIDED OR SELF-DEVELOPED PLUGINS.

The company wants the possibility to create pages to show information about current, former and future events. They also want to publish news items. Make proper use of the provided plugins like SEO, social media support and security.
The website design needs to be classy and modern to fit with the target audience for Global Fashion 2017: Mainly female owners of fashion web shops.

### Website Design Details

Create a design for a responsive front page for the fashion event 'Global Fashion 2017'.
Make a design for how the website will looks like at the following resolutions:
Desktop:1330px,
Tablet:768px,
Mobile: 320px.

Add the designs to a mockup to present your website.

The website consists of 2 page templates. A main page template for the upcoming events, and another page template for former events. You only need to design the main page.

**The main page template:**
The main page template includes
- Page title
- Header video gallery
- Event description (placeholder text)
- Countdown timer (October,19th 18.00)
- News posts
- Book tickets button
- Maps location (Screenshot)
- Tab to open the hidden sidebar
- Footer with copyrights and social media links

**The page template for the former events**
- Page title
- Header video
- Image gallery
- Event description (placeholder text)
- Maps location (Screenshot)
- Tab to open the hidden sidebar
- Footer with copyrights and social media links

The sidebar is added to all pages. The sidebar is hidden and just has a small tab. When clicking the tab, the sidebar slides or fades in.
The sidebar needs to support adding search widget, images of sponsors and a calendar widget.
The footer is added to all pages. The footer contains copyrights and links to at least three social media platforms.

The header section of the main page shows all videos in a gallery –instead of one video:
The video for the 2017 event will cover the header section of the main page. Other videos about former events are displayed smaller. The small videos contain year and place as title.
At the video gallery, when hovering small videos, the hovered video will be played and the large video will be paused. With mouse-out the small video stops and the large video continues.
The video gallery is the eye catcher of the main page. The design must be very appealing, but without loosing the classy and modern look and feel.

The main page shows an accurate countdown timer to the start of the event (October,19th 18.00).

Posts will be displayed in the main page with category icons. You should produce the icons for:
- Fashion catwalk (catwalk show related news items),
- Pop-up stores (news items related to available stores at the venue),
- Brand parade (news items related to the big parade where all famous brands participate)
- Posts will have a short message with 'read more' option and optional there can be an image.

Readability and attraction is very import for the design of the post.

The UI of the slider should have a minimal and elegant design for leading all the attention to the images.

**Style guide Details**
**Necessary elements for the style guide including colors, sizes and fonts**

- Buttons
- Backgrounds
- Main title
- Subpage title
- Section title
- Paragraphs
- Navigation
- News category Icons.

- Add comments to the style guide to specify how to use the logo and the interactive elements.

The designs should be saved as: Desktop_WorldFashion.*, Tablet_WorldFashion.* and Mobile_WorldFashion.*.
The designs should be saved as *.png and .psd or .ai.

Add your designs to one of the provided mockups and save it as:
Mockup_WorldFashion.png
In addition to the website design you need to design a style guide for future development of the website. You have to use one of the two provided logos, but you are allowed to make changes to the color of the logo.
Save your design, style guide and mockup files in this location:
http://competitorYY.wsad.local/XX_cms_module/design
XX is your country code. YY is workstation number.

**TECHNIQUE**

The website layout should be developed with the ability to add menu items (up to 5 items each maximum 15 characters) and content without damaging the design. For future use widgets will be placed in the sidebar. Make sure the content of the sidebar can easily be changed without damaging the design.

The layout of the website needs to be identical to your designs, but also needs to scale without damaging the design when scaling the browser window between 320px and 1330px. Make use of HTML and CSS by the W3C standards for proper SEO support.

Clicking at menu and post items should not cause a page refresh, but post content will be loaded asynchronous from the server.

### CMS Details

For safety reasons two user profiles needs to be created:

- The Admin user - access to the complete WordPress main dashboard.
    - Username: adMinX
    - Password: Never4get!
- The Client user - access to the main WordPress dashboard showing only pages, posts, media and relevant plugins (image sliders, video gallery, security, seo).
    - Username: 4Clients
    - Password: Never4get2

The Wordpress login page should have the logo of GlobalFashion.

**Pages**

In the Wordpress dashboard for each page there are fields to fill for:
- Title
- Place
- Start date (yyyy, mm, dd), end date (yyyy, mm,dd)
- Description of the event
- Location (maps) image
- Video

**Posts**

In the Wordpress dashboard each posts contains:
- Title,
- Date,
- Message,
- Category-icon,

*Images are optional.

**Create Image slider (plugin)**

Create an image slider plugin that appears directly in the main dashboard menu as a plugin.
By using the plugin in the main dashboard menu, clients can create multiple sliders with at least three images. The sliders can be implemented on the pages by adding shortcodes. The slider can be paused by mouse-over, continued by mouse-out and individual images can be shown large by clicking at the corresponding thumbnail or button.

**Header video gallery (plugin)**

Create a plugin that adds a video gallery to the header of the main page. The video gallery contains a large video for the current event and small videos for the past events.
- For all videos year and place of the corresponding event can be described.
- Videos can be uploaded.
- Videos can be removed.
- Videos can be arranged by year (automatically or manual).

*It is allowed to add the code for the video gallery directly into the post page.

# SERVER SIDE

# INTRODUCTION

"**Bani Yas**" is a local startup company. They would like to create a website that could help users to get itinerary of public transportation to reach their destination place by schedule. They will provide the coordinate for train and bus station and a map of Abu Dhabi. The user can set their start and end place/station, then the system will find the fastest route(s) between the two given stations depending on the vehicles/lines running. The routes may include transfers between multiple vehicles/lines. You can use your own algorithm to solve the problem. The system should help the user to decide which route is faster based on the public transportation schedule.

The system should be separated by client and server architecture. The customer is asking for a web service architecture. Development phase should be separated into two phases. The **first phase** is creating the **backend web service** and the **second phase** is creating the **front end**. "**Bani Yas**" has made an initial design of the website to be used for the front-end development phase. You can use and enhance the design that is given with a client side framework to communicate with the services.

Glossary:

Schedule: is a moving from one station/place to the next station/place at specific times.

Line: consists of the multiple schedules and run by a vehicle (for example the any color of the line).

Route: is the trip for a passenger from departure/source to destination/target.

## Description of project and tasks

The description for the first phase of the project is listed below. The first task is to create a restful web service API that can be used by the front end to communicate the data.

**I. Web Service**

"**Bani Yas**" will provide the list of web services that need to be created. Web service specification will contain the URL path of web service, request method, requested parameter on URL, requested parameter on body request, response result and response status. Request and response on web service should only contain JSON.

There are three roles/types of users: public, authenticated user and admin.

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

**1. Authentication**

a. Login (*v1/auth/login*)

Description: For client to get login token via username and password

Request method: **POST**

Header: header authorization basic

Requested parameter:

- Body:
  - Username
  - password

Response result:

- If success,
  - header: response status: 200

- body:
  - token`: authorization token (to be valid until logout). Token will be generated by the system from logged in username with md5 encryption method
  - Role (ADMIN / USER)
- If username/password not correct or empty,
  - header: response status: 401
  - body: message: invalid login

b. Logout (*v1/auth/logout?token={AUTHORIZATION_TOKEN}*)
Description: For server to invalid the user's token
Request method: **GET**
Header: header authorization basic
Response result:
- If success,
  - header: response status: 200
  - body:
    - message: logout success
- If unauthorized user access it, data:
  - Message: Unauthorized user
  - Response status: 401

**2. Place**

a. All Places (*v1/place?token={AUTHORIZATION_TOKEN}*)
Description: For client to list all places in the database (include user's search history indexed based on the frequency)
Request method: **GET**
Header: header authorization basic

Response result:
body:
- All data on array; consists of id, name, latitude, longitude, x, y, image_path, description.
- Response status: 200
- If unauthorized user access it, data:
  - Message: Unauthorized user
  - Response status: 401

b. Find Place (*v1/place/{ID}?token={AUTHORIZATION_TOKEN}*))
Description: For client to fetch one place object via place ID.
Request method: **GET**
Header: header authorization basic
Response result:
- body:
  - object; property consists of id, name, latitude, longitude, x, y, image_path, description.
  - Response status: 200

c. Create place (*v1/place?token={AUTHORIZATION_TOKEN}*), only admin can access this API
Description: For client to create a new place object. Image file from client should be uploaded to server. You can use form data to upload an image.
Request method: **POST**
Header: header authorization basic
Request parameter:
- Body:

- name
- latitude
- longitude
- x
- y
- image
- [description]

Response result:

- If success, body:
  - Message: create success
  - Response status: 200
- If failed, body:
  - Message: Data cannot be processed
  - Response status: 422
- If unauthorized user access it, body:
  - Message: Unauthorized user
  - Response status: 401

d. Delete place (*v1/place/{ID}?token={AUTHORIZATION_TOKEN}*), only admin can access this API

Description: A request to delete a place object via given place ID.

Request method: **DELETE**

Header: header authorization basic

Response result:

- If success, body:
  - Message: delete success
  - Response status: 200
- If failed, body:
  - Message: Data cannot be deleted
  - Response status: 400
- If unauthorized user access it, data:
  - Message: Unauthorized user
  - Response status: 401

e. Update place (*v1/place/{ID}?token={AUTHORIZATION_TOKEN}*), only admin can access this API

Description: For client to update an existing place object via given place ID. If an image file is provided, it should be uploaded to server.

Request method: **POST**

Header: header authorization basic

Request parameter:

- Body:
  - [name]
  - [latitude]
  - [longitude]
  - [x]
  - [y]
  - [image]
  - [description]

Response result:

- If success, body:
  - Message: update success
  - Response status: 200

- If failed, body:
  - Message: Data cannot be updated
  - Response status: 400
- If unauthorized user access it, body:
  - Message: Unauthorized user
  - Response status: 401

**3. Schedule**

a. Create schedule (*v1/schedule?token={AUTHORIZATION_TOKEN}*), only admin can access this API
Description: For client to create a schedule in database. A schedule describes when and where a bus/train departs from one stop and arrive at the next stop.
Request method: **POST**
Header: header authorization basic
Request parameter:
- Body:
  - Object: consisting of type (bus or train), line , from_place_id, to_place_id, departure_time, arrival_time, distance, speed

Response result:
- If success,
  - header: response status: 200
  - body: message: create success
- If failed,
  - header: response status: 422
  - body: message: Data cannot be processed
- If unauthorized user access it,
  - header: response status: 401
  - body: message: Unauthorized user

b. Delete schedule (*v1/schedule/{ID}?token={AUTHORIZATION_TOKEN}*), only admin can access this API
Description: A request to delete an existing schedule via given schedule ID.
Request method: **DELETE**
Header: header authorization basic
Response result:
- If success,
  - header: response status: 200
  - body: message: delete success
- If unauthorized user access it,
  - header: response status: 401
  - body: message: Unauthorized user

**4. Route**

a. Route Search
(*v1/route/search/{FROM_PLACE_ID}/{TO_PLACE_ID}/[DEPARTURE_TIME]?token={AUTHORIZATION_TOKEN}*)
Description: A request to fetch multiple route suggestions to depart from a given stop (departure/source) and arrive at another stop (destination/target). By default, the search uses current server time. It also allows an optional departure time to override the default server time.
The search should search the routes that depart from the given place at specific time and arrive the destination place, sorting by the earliest arrival time and limited to 5 routes result.

The route allows transfer to different bus/train at the same station. All transfer happens at the same stop. There is no walk and no minimum transfer time required.
Request method: **GET**
Header: header authorization basic
Response result:

- If success, data:
  - Array of routes. Each route contains :
    - Number of history selection of this route.
    - Array of schedules:
      - id
      - type
      - line
      - departure_time
      - arrival_time
      - travel_time
      - from_place; consist of id, name, longitude, latitude, x, y, description, image_path
      - to_place; consist of id, name, longitude, latitude, x, y, description, image_path
  - Response status: 200
- If failed, data:
  - Message: Unauthorized user
  - Response status: 401

b. Store Route Selection History (*v1/route/selection?token=[AUTHORIZATION_TOKEN]*)
Description: For client to save a user selected route into the system.
Request method: **POST**
Header: header authorization basic
Request parameter:

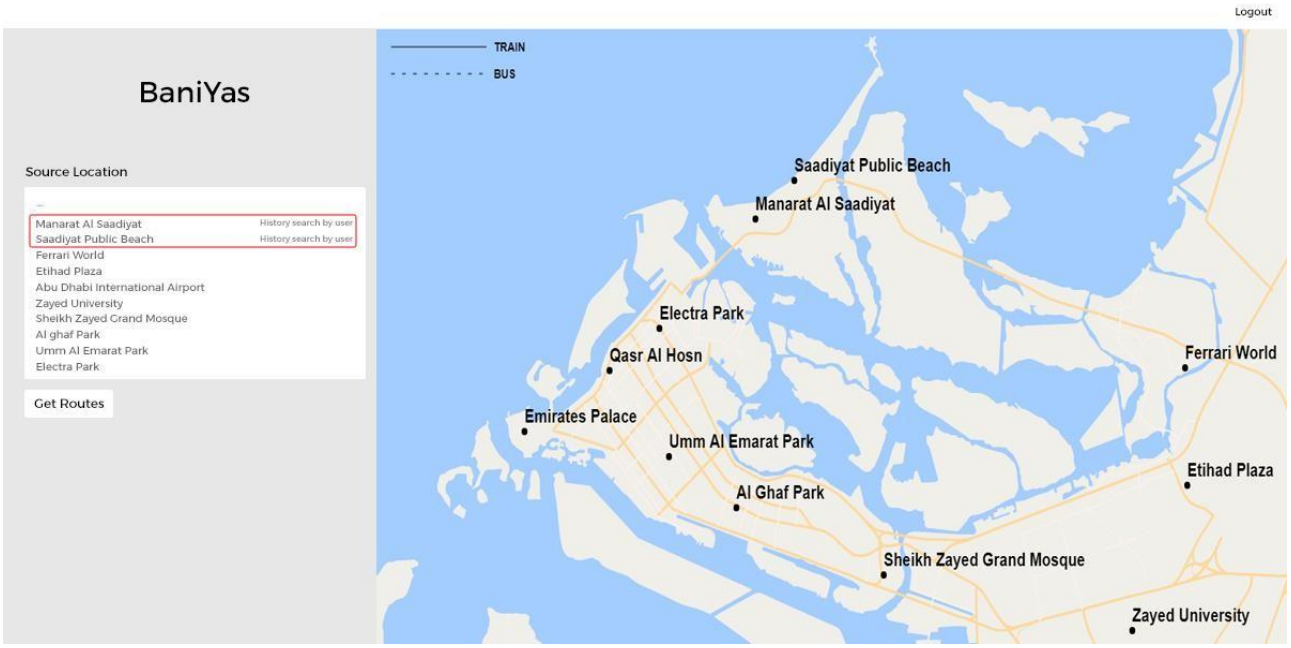
- Body:
  - from_place_id
  - to_place_id
  - schedule_id, array of schedule_id for the route

Response result:

- If success, body:
  - Message: create success
  - Response status: 200
- If failed, body:
  - Message: Data cannot be processed
  - Response status: 422

## II. Front End

On the front-end there will be some functionality that is required by "**Bani Yas**".
Create a front-end website for this company. Create all the functionalities for the page with communication with the backend web services API.




Website components that have been provided are:

1. **Search Route**

   Functionalities:

   a. Selecting *from* place and *to* place
      - Fetch the list of places from the service. Sorted alphabetically by default. Places used by this user (if logged in) will be at top of list (based on frequency).
      - Selection with using auto complete (from database, not from browser history) for the *from (source)* and *to (target)* place. Auto complete will be filled based on user's history, every time user use this application then the system will save the data (*from/source* and *to/target*).

   b. Input departure time
      - Contain hour and minute. This is an optional field, doesn't required to be filled.

   c. Searching routes
      - Search the routes by using the *departure time (optional)*, *from (source)* and *to (target)* input.

2. **Routes List**

   After a user search the routes, then this component will show:
   Search results on the left, schedules for this route will appear. By default, the search uses the current server time if user didn't fill in *departure time* before. If the user fills in departure time, then the search uses the given *departure time*.
   As same as the API specification describes: The search should search the routes that depart from the given place at specific time and arrive to the destination place, sorting by the earliest arrival time and limited to 5 routes result.
   The route allows transfer to different bus/train at the same station. A transfer happens at the same stop. There is no walk and no minimum transfer time required.

   The list of train or bus schedule from the result, consist of:
   - Numbering.
   - Time Schedule **(Departure time at from place, Arrival time at to place)**.
   - Total travel time
   - Number of transfers/changes of line

- Number of selection on this routes by all users (more on that later in 4. Search history).
- This is the example scenario for search results, assuming from A to B that departs at 13:00:
  - A to B => departure time -> 13:00:00 arrival time -> 13:14:00, Bus Line 3, 14 minutes, 0 transfers.
  - A to B => departure time -> 13:02:00 arrival time -> 13:22:00, Bus Line 2, Train Line 4, 22 minutes, 1 transfer
  - A to B => departure time -> 13:01:00 arrival time -> 13:35:00, Bus Line 2, Bus Line 1, Bus Line 5, 36 minutes, 2 transfer

**3. Map View**

The right side of the layout will show the map. Place coordinate will contain:

a. There will be dot and place's name on map for each place based on the database records.
b. If user clicks one of the dot, a floating box will be appeared near the dot. On floating box there will be a picture of the place, name of place and the short description about it. The box will be dismissed if user clicks the other area.
c. Show a clearly visible dot (different from the normal place) on map for departure (from) place and destination (to) place.
d. If user clicks the route on the result list, the route is shown on the map. Solid line for train, dashed line for bus.
e. Each bus/train line should have a different color when drawing on the map.
f. Show the legend for each vehicle line: different color for each line, and solid line for train, dashed line for bus.

**4. Route(s) selection history**

The system stores all the routes that all user selected.

a. Whenever a user clicks on the route in the result list, this route is stored in the database.
b. The system stores all user's selection.
c. When a route shows in the result list, the total number of selection on the same route shows. Same route means the same from place and to place, regardless of departure time.

5. **User Authentication**

Functionalities:

a. Login and logout should happen on the same page without redirect.
b. Login
   - Show the login modal, after user click login link.
   - On the login dialog there will be inputs for username and password.
   - After the user logged in, the login link will be changed to logout link and the current username will be displayed besides the logout link.
   - There will be role for the two types of authenticated user: if user is an admin the admin menu will be shown.
   - The username entered and token received, token will be kept on the client for further requests, also after page refresh.
c. Logout
   - The display is reset: login link is shown, username and corresponding functionality disappear.

6. **Admin Menu**
   Admin menu should only be shown after user login that has an admin role. Admin functionalities:
   a. Place
      - Create place
      - Update place
      - Delete place

**Notes**

- Competitors should implement a minimum of one of the server-side and client-side frameworks/libraries that are provided.
- The provided template design should be used, but it can be enhanced to get better functionality for your site.
- Show error/feedback messages based on response from API.
- The specified database tables need to be implemented. More tables may be added if needed. Provide a final SQL-dump and ERD screen as specified below.
  All API should fulfill all requirements as stated in the description. All prefix, RESTful-URL and HTTP-Method from given API link should be implemented correctly and not be changed. If needed, you may add other API, besides all API that already mentioned in this document.
- Create the following users to login to the system:
  - Admin with username: `admin` and password: `adminpass`,
  - User1 with username: `user1` and password: `user1pass`,
  - User2 with username: `user2` and password: `user2pass`
- Changes made in the data on the back-end server need to be propagated to the frontend. The data should be dynamically shown.
- Monitor screen size should not ***affect*** any function on the client side. (working on 1440px, 768px and 320px width)

# CLIENT SIDE

## Introduction

In recent years, the internet has become an integral part of our daily lives, enabling the dissemination of information in an inexhaustible source of content and interaction. Every day the use of games has gained a prominent role in this universe, allowing millions of people to get access to fun and entertainment quickly and free.

Thinking about these concepts, you decided to develop a small game that works in the most common web browsers and that makes it possible to spread your talent in the skill of web design and development. The game will be called **Star Battle**.

You should design the game, develop the layout using HTML and CSS and develop client-side programming using JavaScript and its open source libraries. Some media files are available to you in a zip file. You can create more media and modify anything in the media if you want.

To be used in different resolutions your game needs to be developed in a tablet resolution with 960x600 pixels. But, if game is open in a big screen, the game must be in the center of the screen (horizontally and vertically).

## Description of project and tasks

This is a module of 5 hours. Your first 2.5 hours must be used to create the design of the game in three PNG images and the initial layout using HTML/CSS. Your layout should follow the design that you created. The final 2.5 hours you will create the functionality of game using JavaScript that allows the game to work correctly in different web browsers, following the requirements described below.

Star Battle game uses elements described below:

1. Main spaceship: Element that is controlled by the player.
2. Planets in background: Elements that move from right to left to give the impression of movement of the spaceship in the space.
3. Enemy spaceships: Elements that player needs to destroy to get points.
4. Asteroids: Elements that player needs to destroy to get points.
5. Friendly spaceships: Elements that player shouldn't destroy or player will lose points.
6. Fuel icon: Elements that player needs to collect to increment the fuel level.
7. Fuel Counter: Element that shows how much fuel is available. It must be a number and a graphical element and it should be animated when the fuel is decreasing and when the user gets more fuel.
8. Score Counter: Elements that shows how many points the user got destroying asteroids and enemy's spaceships.
9. Timer: Element that shows how much time the spaceship is flying.
10. Font Size Buttons: Elements that increase and decrease the font size.
11. Pause/Continue button: Button to pause/continue the game.
12. Sensible areas to control the spaceship.
13. Logo: Add the provided logo in the game.
14. Shot: Shot by the ships.

**FIRST 2.5 HOURS – DESIGN AND INITIAL LAYOUT:**

1. **Deliver at least 3 PNG image files that present:**
   1.1. Game Instructions: The first screen of the game presents the instructions to the user and the "Start Game" button. The instructions for the game are included in the media files.
   1.2. Game board layout: This design must present all 15 elements described above in the game screen.
   1.3. Ranking Table presentation: This design must present the logo of game and ranking with the following columns: position, name, score and time in this order, with the table is presented the "Start Game" button.

2. Develop the initial markup (HTML + CSS) of your game application. The game is presented to the user with the game instructions and the button "Start Game". The instructions must be presented in an animated way.

3. "Start Game" buttons must have active and hover effects. The background of the buttons in hover state must be: #f19e0d. The active state must follow the example called ripple which is provided in the media files.

4. You should draw the elements described below to be included in your design. Create the elements that represent the same visual style.
   4.1. Main spaceship (controlled by player)
   4.2. Timer Icon
   4.3. Font size buttons
   4.4. Pause/continue button
   4.5. Fuel counter
   4.6. Fuel icon

5. The HTML and CSS code must be valid in the W3C standards for HTML 5 and CSS 3 rules.