LAMPIRAN

A. Code

1. Ambil.php

```
<?php
include 'klien.php';
include 'EasyVars.php';
include 'setlokasiDB.php';
include 'setlokasi.php';

$servername = "localhost";
$username = "botjadwaltele_DB";
$password = "Persida87";
$dbname = "botjadwaltele_BOTKU";

$client = new Client("762001393:AAEre4L7mq7I0AyuYigIxJR_xtQDl8tuZRA");
$inline_keyboard = [
   ['text' => 'Set lokasi', 'request_location' => true],
   ['text' => 'Metode Kalkulasi'],
   ['text' => 'Jadwal sholat'],
   ['text' => 'info']
] ;
$inline_key = [
   ['text' => 'Berita Baru', 'url' => 'https://suryaformosa.com/category/headline-news/'],
   ['text' => 'Tentang PCIM Taiwan', 'url' => 'https://www.youtube.com/watch?v=2e-qm1tPeOI&t=3s']
] ;

$okeee = array('inline_keyboard' =>
      array(
         array(
            array(
               'text' => "Jafari, Ithna Ashari", 'callback_data' => '0'
            )
         ),
         array(
            array(
               'text' => "Karachi, University Of Islamic Sciences", 'callback_data' => '1'
            )
         ),
         array(
            array(
               'text' => "ISNA, Islamic Society Of North America", 'callback_data' => '2'
```

```
            )
        ),
        array(
            array(
                'text' => "MWL, Muslim World League", 'callback_data' => '3'
            )
        ),
        array(
            array(
                'text' => "Makkah, Umm Al-Qura", 'callback_data' => '4'
            )
        ),
        array(
            array(
                'text' => "Egypt, Egyptian General Authority Of Survey", 'callback_data' => '5'
            )
        ),
        array(
            array(
                'text' => "University Of Tehran, Institute Of Geophysic", 'callback_data' => '6'
            )
        )
    )
```

```
    );
// METODE KALKULASI
$arraybiasa = [0,1,2,3,4,5,6,7];
$arraymetode = ["Jafari, Ithna Ashari",
        "Karachi, University Of Islamic Sciences",
        "ISNA, Islamic Society Of North America",
        "MWL, Muslim World League",
        "Makkah, Umm Al-Qura",
        "Egypt, Egyptian General Authority Of Survey",
        "null",
        "University Of Tehran, Institute Of Geophysic"];


$kata = "Assalamu'alaikum warahmatullah wabarokatuh \n \nSelamat datang di Muhammadiyah Taiwan Bot,yang akan membantu Anda mengakses jadwal waktu sholat sesuai lokasi Anda. \n \n*Klik Set Lokasi untuk segera mengaktifkan penghitungan jadwal waktu sholat*. \n \nSelamat menikmati layanan kami. \n \nWassalamu'alaikum
```

```
warahmatullah wabarokatuh. \nTim LIKKI PCIM Taiwan";
$offset=0;
while (true) {
  $updates = $client->getUpdates($offset);
  if ($updates->ok == true) {
    foreach ($updates->result as $update) {
      $offset = $update->update_id + 1;
      $easy = new EasyVars($update);
      // ini callback
      if (isset($update->callback_query)){
        $callback_query = $update->callback_query;
        $callback_queryId = $callback_query->id;
        $callback_queryUserId = $callback_query->from->id;
        $callback_queryData = $callback_query->data;
        $hapus_Call = $callback_query->message->message_id;


switch($callback_queryData){
          case '0':
            $conn = new mysqli($servername, $username, $password,$dbname);
            $sql = "UPDATE userProfile SET metode=".$arraybiasa[0]." WHERE chat_id=".$chat_id;
            $conn->query($sql);
            $conn->close();
            $client->answerCallbackQuery($callback_queryId,"Pemilihan metode Jafari, Ithna Ashari Diterima", true);
            $client->deleteMessage($chat_id,$hapus_Call);
            $client->sendMessage($chat_id, "Silahkan lihat kembali jadwal shalat");
            break;

          case '1':
            $conn = new mysqli($servername, $username, $password,$dbname);
            $sql = "UPDATE userProfile SET metode=".$arraybiasa[1]." WHERE chat_id=".$chat_id;
            $conn->query($sql);
```

```
                $conn->close();
                $client->answerCallbackQuery($callback_queryId, "Pemilihan metode Karachi, University Of Islamic Sciences Diterima", true);
                $client->deleteMessage($chat_id,$hapus_Call);
                $client->sendMessage($chat_id, "Silahkan lihat kembali jadwal shalat");
                break;

                case '2':
                $conn = new mysqli($servername, $username, $password,$dbname);
                $sql = "UPDATE userProfile SET metode=".$arraybiasa[2]." WHERE chat_id=".$chat_id;
                $conn->query($sql);
                $conn->close();
                $client->answerCallbackQuery($callback_queryId, "Pemilihan metode ISNA, Islamic Society Of North America Diterima", true);
                $client->deleteMessage($chat_id,$hapus_Call);
                $client->sendMessage($chat_id, "Silahkan lihat kembali jadwal shalat");
                break;

                case '3':
                $conn = new mysqli($servername, $username, $password,$dbname);
                $sql = "UPDATE userProfile SET metode=".$arraybiasa[3]." WHERE chat_id=".$chat_id;
                $conn->query($sql);
                $conn->close();
                $client->answerCallbackQuery($callback_queryId, "Pemilihan metode MWL, Muslim World League Diterima", true);
                $client->deleteMessage($chat_id,$hapus_Call);
                $client->sendMessage($chat_id, "Silahkan lihat kembali jadwal shalat");
                break;
```

```
                case '4':
                $conn = new mysqli($servername, $username, $password,$dbname);
                $sql = "UPDATE userProfile SET metode=".$arraybiasa[4]." WHERE chat_id=".$chat_id;
                $conn->query($sql);
                $conn->close();
                $client->answerCallbackQuery($callback_queryId, "Pemilihan metode Makkah, Umm Al-Qura Diterima", true);
                $client->deleteMessage($chat_id,$hapus_Call);
                $client->sendMessage($chat_id, "Silahkan lihat kembali jadwal shalat");
                break;

                case '5':
                $conn = new mysqli($servername, $username, $password,$dbname);
                $sql = "UPDATE userProfile SET metode=".$arraybiasa[5]." WHERE chat_id=".$chat_id;
                $conn->query($sql);
                $conn->close();
                $client->answerCallbackQuery($callback_queryId, "Pemilihan metode Egypt, Egyptian General Authority Of Survey Diterima", true);
                $client->deleteMessage($chat_id,$hapus_Call);
                $client->sendMessage($chat_id, "Silahkan lihat kembali jadwal shalat");
                break;

                case '6':
                $conn = new mysqli($servername, $username, $password,$dbname);
                $sql = "UPDATE userProfile SET metode=".$arraybiasa[7]." WHERE chat_id=".$chat_id;
                $conn->query($sql);
                $conn->close();
                $client->answerCallbackQuery($callback_queryId, "Pemilihan metode University Of Tehran, Institute Of Geophysic Diterima", true);
```

```
            $client->deleteMessage($chat_id,$hapus_Call);
            $client->sendMessage($chat_id, "Silahkan lihat kembali jadwal shalat");
            break;
        }

    }
    //ini pesan
    if (isset($update->message)) {
        $chat_id = $update->message->chat->id;


        if (isset($update->message->reply_to_message->from->id)) {
            $reply_id = $update->message->reply_to_message->from->id;
        }
        //atur lokasi
        if(isset($update->message->location->longitude)){
            $longitude=$update->message->location->longitude;
            $latitude=$update->message->location->latitude;
```

```
            $gmt = TimezoneMapper::latLngToTimezoneValue($latitude,$longitude);
            $gmtstring = TimezoneMapper::latLngToTimezoneString($latitude,$longitude);
            $conn = new mysqli($servername, $username, $password,$dbname);
            $sql = "UPDATE userProfile SET lat=".$latitude.",lon=".$longitude." ,timezone=".$gmt.", metode=".$arraybiasa[4]." WHERE chat_id=".$chat_id;
            $conn->query($sql);
            $conn->close();
            $client->sendMessage($chat_id, "Saat ini lokasi anda berada di \nLintang: ".$longitude." \nBujur: ".$latitude." \nZona Waktu: ".$gmtstring." \n\nJadwal Waktu Sholat anda telah diatur sesuai lokasi diatas, jika anda bepergian keluar kota/tempat lain, mohon segera atur ulang lokasi anda.\n\n silahkan klik button metode kalkulasi untuk menentukan metode.");
```

```
            }

            $text = $easy->text;
            switch($text){
                case '/start':
                $client->sendMessage($chat_id, $kata,'Markdown',false,false,0, ['keyboard' => [$inline_keyboard], 'resize_keyboard'=>true]);
                //======= conect db
                $conn = new mysqli($servername, $username, $password,$dbname);

                $sql = "SELECT * FROM userProfile WHERE chat_id=".$chat_id;
                $result = $conn->query($sql);
                $row = $result->fetch_assoc();

if($row["chat_id"]==$chat_id){
                    $conn->close();
                }else{
                    $sql1 = "INSERT INTO userProfile (chat_id) VALUES ('".$chat_id."')";
                    $conn->query($sql1);
                    $conn->close();
                }
                break;

                case 'Jadwal sholat':
                // read data base
                $conn = new mysqli($servername, $username, $password,$dbname);
                $sql = "SELECT lat,lon,timezone,metode FROM userProfile WHERE chat_id=".$chat_id;
                $result = $conn->query($sql);
                $row = $result->fetch_assoc();


$latitudedb=$row["lat"];

$longitudedb=$row["lon"];

$timeZonedb=$row["timezone"];

$timeZonedb=intval($row["timezone"])>=0?"+".$row["timezone"]:$row["timezone"];

$metodedb=$row["metode"];
                $conn->close();
```

```
                $method=$metodedb;
                //$date=date("Y-M-
d");
                $prayTime = new
PrayTime($method);
                $times = $prayTime-
>getPrayerTimes(time(),
$latitudedb, $longitudedb,
$timeZonedb, $method);
                $client-
>sendMessage($chat_id,
"JADWAL SHOLAT HARI INI
\n\nShubuh \n----------
".$times[0]."\nMatahari terbit \n---
--------------------
".$times[1]."\nDhuhur \n----------
---------- ".$times[2]."\nAshar \n--
--------------------
".$times[3]."\nMaghrib \n----------
---------- ".$times[5]."\nIsya' \n---
--------------------
".$times[6]."\n\nGMT".$timeZone
db." \nKoordinat: ".$latitudedb.",
".$longitudedb." \nMetode
Kalkulasi:
".$arraymetode[$method]);
                break;

                case 'Metode
Kalkulasi':
                $client-
>sendMessage($chat_id, "*
Silahkan Pilih Metode Kalkulasi
:*",

'Markdown',false,false,0,$okeee);
                break;

                case 'info':
                $client-
>sendMessage($chat_id, "*Info
Tentang Pimpinan Cabang
Istimewa Muhammadiyah
Taiwan*",

'Markdown',false,false,0,['inline_k
eyboard' => [$inline_key]]);
                break;

                default:
                // $client-
>sendMessage($chat_id, $easy-
>message_id);
                break;
            }

        }

    }
  } else {
    exit($updates->description);
```

```
    }
```

```
}
```

2. API.php

```
<?php /** @noinspection PhpOptionalBeforeRequiredParametersInspection */
//functions automatically generated from https://core.telegram.org/bots/api
// namespace TuriBot;
include 'ApiInterface.php';
abstract Class Api implements ApiInterface
{
    public function getUpdates(
        int $offset = null,
        int $limit = null,
        int $timeout = null,
        array $allowed_updates = null
    ) {
        $args = [];
        if ($offset !== null) {
            $args['offset'] = $offset;
        }
        if ($limit !== null) {
            $args['limit'] = $limit;
        }
        if ($timeout !== null) {
            $args['timeout'] = $timeout;
        }
        if ($allowed_updates !== null) {
            $args['allowed_updates'] = $allowed_updates;
        }
        return $this->Request('getUpdates', $args);
    }
    public function setWebhook(
        string $url,
        \CURLFile $certificate = null,
        int $max_connections = null,
        array $allowed_updates = null
    ) {
        $args = [
            'url' => $url
        ];
        if ($certificate !== null) {
            $args['certificate'] = $certificate;
        }
        if ($max_connections !== null) {
            $args['max_connections'] = $max_connections;
```

```
        }
        if ($allowed_updates !== null) {
            $args['allowed_updates'] = $allowed_updates;
        }
        return $this->Request('setWebhook', $args);
    }
    public function deleteWebhook()
    {
        return $this->Request('deleteWebhook', []);
    }
    public function getWebhookInfo()
    {
        return $this->Request('getWebhookInfo', []);
    }
    public function getMe()
    {
        return $this->Request('getMe', []);
    }
    public function sendMessage(
        $chat_id,
        string $text,
        string $parse_mode = null,
        bool $disable_web_page_preview = null,
        bool $disable_notification = null,
        int $reply_to_message_id = null,
        array $reply_markup = null
    ) {
        $args = [
            'chat_id' => $chat_id,
            'text'    => $text
        ];
        if ($parse_mode !== null) {
            $args['parse_mode'] = $parse_mode;
        }
        if ($disable_web_page_preview !== null) {
            $args['disable_web_page_preview'] = $disable_web_page_preview;
        }
        if ($disable_notification !== null) {
            $args['disable_notification'] = $disable_notification;
        }
```

```
        if ($reply_to_message_id !== null) {
            $args['reply_to_message_id'] = $reply_to_message_id;
        }
        if ($reply_markup !== null) {
            $args['reply_markup'] = json_encode($reply_markup);
        }
        return $this->Request('sendMessage', $args);
    }
    public function forwardMessage(
        $chat_id,
        $from_chat_id,
        bool $disable_notification = null,
        int $message_id
    ) {
        $args = [
            'chat_id'      => $chat_id,
            'from_chat_id' => $from_chat_id,
            'message_id'   => $message_id
        ];
        if ($disable_notification !== null) {
            $args['disable_notification'] = $disable_notification;
        }
        return $this->Request('forwardMessage', $args);
    }
    public function sendPhoto(
        $chat_id,
        $photo,
        string $caption = null,
        string $parse_mode = null,
        bool $disable_notification = null,
        int $reply_to_message_id = null,
        array $reply_markup = null
    ) {
        $args = [
            'chat_id' => $chat_id,
            'photo'   => $photo
        ];
        if ($caption !== null) {
            $args['caption'] = $caption;
        }
        if ($parse_mode !== null) {
            $args['parse_mode'] = $parse_mode;
        }
```

```
        if ($disable_notification !== null) {

$args['disable_notification'] = $disable_notification;
        }
        if ($reply_to_message_id !== null) {

$args['reply_to_message_id'] = $reply_to_message_id;
        }
        if ($reply_markup !== null) {
            $args['reply_markup'] = json_encode($reply_markup);
        }
        return $this->Request('sendPhoto', $args);
    }
    public function sendAudio(
        $chat_id,
        $audio,
        string $caption = null,
        string $parse_mode = null,
        int $duration = null,
        string $performer = null,
        string $title = null,
        $thumb = null,
        bool $disable_notification = null,
        int $reply_to_message_id = null,
        array $reply_markup = null
    ) {
        $args = [
            'chat_id' => $chat_id,
            'audio'   => $audio
        ];
        if ($caption !== null) {
            $args['caption'] = $caption;
        }
        if ($parse_mode !== null) {
            $args['parse_mode'] = $parse_mode;
        }
        if ($duration !== null) {
            $args['duration'] = $duration;
        }
        if ($performer !== null) {
            $args['performer'] = $performer;
        }
        if ($title !== null) {
            $args['title'] = $title;
        }
        if ($thumb !== null) {
            $args['thumb'] = $thumb;
        }
        if ($disable_notification !== null) {
```

```
$args['disable_notification'] = $disable_notification;
        }
        if ($reply_to_message_id !== null) {
            $args['reply_to_message_id'] = $reply_to_message_id;
        }
        if ($reply_markup !== null) {
            $args['reply_markup'] = json_encode($reply_markup);
        }
        return $this->Request('sendAudio', $args);
    }
    public function sendDocument(
        $chat_id,
        $document,
        $thumb = null,
        string $caption = null,
        string $parse_mode = null,
        bool $disable_notification = null,
        int $reply_to_message_id = null,
        array $reply_markup = null
    ) {
        $args = [
            'chat_id' => $chat_id,
            'document' => $document
        ];
        if ($thumb !== null) {
            $args['thumb'] = $thumb;
        }
        if ($caption !== null) {
            $args['caption'] = $caption;
        }
        if ($parse_mode !== null) {
            $args['parse_mode'] = $parse_mode;
        }
        if ($disable_notification !== null) {
            $args['disable_notification'] = $disable_notification;
        }
        if ($reply_to_message_id !== null) {
            $args['reply_to_message_id'] = $reply_to_message_id;
        }
        if ($reply_markup !== null) {
            $args['reply_markup'] = json_encode($reply_markup);
        }
        return $this->Request('sendDocument', $args);
    }
```

```
    public function sendVideo(
        $chat_id,
        $video,
        int $duration = null,
        int $width = null,
        int $height = null,
        $thumb = null,
        string $caption = null,
        string $parse_mode = null,
        bool $supports_streaming = null,
        bool $disable_notification = null,
        int $reply_to_message_id = null,
        array $reply_markup = null
    ) {
        $args = [
            'chat_id' => $chat_id,
            'video'   => $video
        ];
        if ($duration !== null) {
            $args['duration'] = $duration;
        }
        if ($width !== null) {
            $args['width'] = $width;
        }
        if ($height !== null) {
            $args['height'] = $height;
        }
        if ($thumb !== null) {
            $args['thumb'] = $thumb;
        }
        if ($caption !== null) {
            $args['caption'] = $caption;
        }
        if ($parse_mode !== null) {
            $args['parse_mode'] = $parse_mode;
        }
        if ($supports_streaming !== null) {
            $args['supports_streaming'] = $supports_streaming;
        }
        if ($disable_notification !== null) {

            $args['disable_notification'] = $disable_notification;
        }
        if ($reply_to_message_id !== null) {

            $args['reply_to_message_id'] = $reply_to_message_id;
        }
        if ($reply_markup !== null) {
            $args['reply_markup'] = json_encode($reply_markup);
        }
```

```
        return $this->Request('sendVideo', $args);
    }
    public function sendAnimation(
        $chat_id,
        $animation,
        int $duration = null,
        int $width = null,
        int $height = null,
        $thumb = null,
        string $caption = null,
        string $parse_mode = null,
        bool $disable_notification = null,
        int $reply_to_message_id = null,
        array $reply_markup = null
    ) {
        $args = [
            'chat_id'   => $chat_id,
            'animation' => $animation
        ];
        if ($duration !== null) {
            $args['duration'] = $duration;
        }
        if ($width !== null) {
            $args['width'] = $width;
        }
        if ($height !== null) {
            $args['height'] = $height;
        }
        if ($thumb !== null) {
            $args['thumb'] = $thumb;
        }
        if ($caption !== null) {
            $args['caption'] = $caption;
        }
        if ($parse_mode !== null) {
            $args['parse_mode'] = $parse_mode;
        }
        if ($disable_notification !== null) {
            $args['disable_notification'] = $disable_notification;
        }
        if ($reply_to_message_id !== null) {
            $args['reply_to_message_id'] = $reply_to_message_id;
        }
        if ($reply_markup !== null) {
            $args['reply_markup'] = json_encode($reply_markup);
        }
        return $this->Request('sendAnimation', $args);
    }
    public function sendVoice(
```

```
        $chat_id,
        $voice,
        string $caption = null,
        string $parse_mode = null,
        int $duration = null,
        bool $disable_notification = null,
        int $reply_to_message_id = null,
        array $reply_markup = null
    ) {
        $args = [
            'chat_id' => $chat_id,
            'voice'   => $voice
        ];
        if ($caption !== null) {
            $args['caption'] = $caption;
        }
        if ($parse_mode !== null) {
            $args['parse_mode'] = $parse_mode;
        }
        if ($duration !== null) {
            $args['duration'] = $duration;
        }
        if ($disable_notification !== null) {

$args['disable_notification'] = $disable_notification;
        }
        if ($reply_to_message_id !== null) {

$args['reply_to_message_id'] = $reply_to_message_id;
        }
        if ($reply_markup !== null) {
            $args['reply_markup'] = json_encode($reply_markup);
        }
        return $this->Request('sendVoice', $args);
    }
    public function sendVideoNote(
        $chat_id,
        $video_note,
        int $duration = null,
        int $length = null,
        $thumb = null,
        bool $disable_notification = null,
        int $reply_to_message_id = null,
        array $reply_markup = null
    ) {
        $args = [
            'chat_id'    => $chat_id,
            'video_note'  => $video_note
        ];
```

```
        if ($duration !== null) {
            $args['duration'] =
$duration;
        }
        if ($length !== null) {
            $args['length'] = $length;
        }
        if ($thumb !== null) {
            $args['thumb'] = $thumb;
        }
        if ($disable_notification !==
null) {

$args['disable_notification'] =
$disable_notification;
        }
        if ($reply_to_message_id !==
null) {

$args['reply_to_message_id'] =
$reply_to_message_id;
        }
        if ($reply_markup !== null) {
            $args['reply_markup'] =
json_encode($reply_markup);
        }
        return $this-
>Request('sendVideoNote', $args);
    }
    public function
sendMediaGroup(
        $chat_id,
        $media,
        bool $disable_notification =
null,
        int $reply_to_message_id =
null
    ) {
        $args = [
            'chat_id' => $chat_id,
            'media'   => $media
        ];
        if ($disable_notification !==
null) {

$args['disable_notification'] =
$disable_notification;
        }
        if ($reply_to_message_id !==
null) {

$args['reply_to_message_id'] =
$reply_to_message_id;
        }
        return $this-
>Request('sendMediaGroup',
$args);
    }
    public function sendLocation(
        $chat_id,
        float $latitude,
        float $longitude,
```

```
        int $live_period = null,
        bool $disable_notification = null,
        int $reply_to_message_id = null,
        array $reply_markup = null
    ) {
        $args = [
            'chat_id'   => $chat_id,
            'latitude'  => $latitude,
            'longitude' => $longitude
        ];
        if ($live_period !== null) {
            $args['live_period'] = $live_period;
        }
        if ($disable_notification !== null) {
            $args['disable_notification'] = $disable_notification;
        }
        if ($reply_to_message_id !== null) {
            $args['reply_to_message_id'] = $reply_to_message_id;
        }
        if ($reply_markup !== null) {
            $args['reply_markup'] = json_encode($reply_markup);
        }
        return $this->Request('sendLocation', $args);
    }
    public function editMessageLiveLocation(
        $chat_id = null,
        int $message_id = null,
        string $inline_message_id = null,
        float $latitude,
        float $longitude,
        array $reply_markup = null
    ) {
        $args = [
            'latitude'  => $latitude,
            'longitude' => $longitude
        ];
        if ($chat_id !== null) {
            $args['chat_id'] = $chat_id;
        }
        if ($message_id !== null) {
            $args['message_id'] = $message_id;
        }
        if ($inline_message_id !== null) {
            $args['inline_message_id'] = $inline_message_id;
        }
        if ($reply_markup !== null) {
```

```
            $args['reply_markup'] =
json_encode($reply_markup);
        }
        return $this-
>Request('editMessageLiveLocati
on', $args);
    }
    public function
stopMessageLiveLocation(
        $chat_id = null,
        int $message_id = null,
        string $inline_message_id =
null,
        array $reply_markup = null
    ) {
        $args = [];
        if ($chat_id !== null) {
            $args['chat_id'] = $chat_id;
        }
        if ($message_id !== null) {
            $args['message_id'] =
$message_id;
        }
        if ($inline_message_id !==
null) {
            $args['inline_message_id']
= $inline_message_id;
        }
        if ($reply_markup !== null) {
            $args['reply_markup'] =
json_encode($reply_markup);
        }
        return $this-
>Request('stopMessageLiveLocati
on', $args);
    }
    public function sendVenue(
        $chat_id,
        float $latitude,
        float $longitude,
        string $title,
        string $address,
        string $foursquare_id = null,
        string $foursquare_type =
null,
        bool $disable_notification =
null,
        int $reply_to_message_id =
null,
        array $reply_markup = null
    ) {
        $args = [
            'chat_id'   => $chat_id,
            'latitude'  => $latitude,
            'longitude' => $longitude,
            'title'     => $title,
            'address'   => $address
        ];
        if ($foursquare_id !== null) {
            $args['foursquare_id'] =
$foursquare_id;
        }
```

```
        if ($foursquare_type !== null)
{
            $args['foursquare_type']   =
$foursquare_type;
        }
        if   ($disable_notification   !==
null) {

$args['disable_notification']          =
$disable_notification;
        }
        if ($reply_to_message_id !==
null) {

$args['reply_to_message_id']       =
$reply_to_message_id;
        }
        if ($reply_markup !== null) {
            $args['reply_markup']      =
json_encode($reply_markup);
        }
        return                          $this-
>Request('sendVenue', $args);
    }
    public function sendContact(
        $chat_id,
        string $phone_number,
        string $first_name,
        string $last_name = null,
        string $vcard = null,
        bool   $disable_notification   =
null,
        int   $reply_to_message_id   =
null,
        array $reply_markup = null
    ) {
        $args = [
            'chat_id'      => $chat_id,
            'phone_number'             =>
$phone_number,
            'first_name'               =>
$first_name
        ];
        if ($last_name !== null) {
            $args['last_name']         =
$last_name;
        }
        if ($vcard !== null) {
            $args['vcard'] = $vcard;
        }
        if   ($disable_notification   !==
null) {

$args['disable_notification']          =
$disable_notification;
        }
        if ($reply_to_message_id !==
null) {

$args['reply_to_message_id']       =
$reply_to_message_id;
```

```
        }
        if ($reply_markup !== null) {
            $args['reply_markup']      =
json_encode($reply_markup);
        }
        return                          $this-
>Request('sendContact', $args);
    }
    public function sendPoll(
        $chat_id,
        string $question,
        array $options,
        bool  $disable_notification  =
null,
        int   $reply_to_message_id   =
null,
        array $reply_markup = null
    ) {
        $args = [
            'chat_id'  => $chat_id,
            'question' => $question,
            'options'  => $options
        ];
        if  ($disable_notification  !==
null) {

$args['disable_notification']       =
$disable_notification;
        }
        if ($reply_to_message_id !==
null) {

$args['reply_to_message_id']      =
$reply_to_message_id;
        }
        if ($reply_markup !== null) {
            $args['reply_markup']      =
json_encode($reply_markup);
        }
        return                          $this-
>Request('sendPoll', $args);
    }
    public function sendChatAction(
        $chat_id,
        string $action
    ) {
        $args = [
            'chat_id' => $chat_id,
            'action'  => $action
        ];
        return                          $this-
>Request('sendChatAction',
$args);
    }
    public                          function
getUserProfilePhotos(
        int $user_id,
        int $offset = null,
        int $limit = null
    ) {
        $args = [
            'user_id' => $user_id
```

```
        ];
        if ($offset !== null) {
            $args['offset'] = $offset;
        }
        if ($limit !== null) {
            $args['limit'] = $limit;
        }
        return $this->Request('getUserProfilePhotos', $args);
    }
    public function getFile(
        string $file_id
    ) {
        $args = [
            'file_id' => $file_id
        ];
        return $this->Request('getFile', $args);
    }
    public function kickChatMember(
        $chat_id,
        int $user_id,
        int $until_date = null
    ) {
        $args = [
            'chat_id' => $chat_id,
            'user_id' => $user_id
        ];
        if ($until_date !== null) {
            $args['until_date'] = $until_date;
        }
        return $this->Request('kickChatMember', $args);
    }
    public function unbanChatMember(
        $chat_id,
        int $user_id
    ) {
        $args = [
            'chat_id' => $chat_id,
            'user_id' => $user_id
        ];
        return $this->Request('unbanChatMember', $args);
    }
    public function restrictChatMember(
        $chat_id,
        int $user_id,
        array $permissions,
        int $until_date = null
    ) {
        $args = [
            'chat_id'     => $chat_id,
            'user_id'     => $user_id,
```

```
            'permissions' =>
json_encode($permissions)
        ];
        if ($until_date !== null) {
            $args['until_date'] =
$until_date;
        }
        return $this-
>Request('restrictChatMember',
$args);
    }
    public function
promoteChatMember(
        $chat_id,
        int $user_id,
        bool $can_change_info = null,
        bool $can_post_messages =
null,
        bool $can_edit_messages =
null,
        bool $can_delete_messages =
null,
        bool $can_invite_users = null,
        bool $can_restrict_members =
null,
        bool $can_pin_messages =
null,
        bool $can_promote_members
= null
    ) {
        $args = [
            'chat_id' => $chat_id,
            'user_id' => $user_id
        ];
        if ($can_change_info !==
null) {
            $args['can_change_info'] =
$can_change_info;
        }
        if ($can_post_messages !==
null) {
            $args['can_post_messages']
= $can_post_messages;
        }
        if ($can_edit_messages !==
null) {
            $args['can_edit_messages']
= $can_edit_messages;
        }
        if ($can_delete_messages !==
null) {

$args['can_delete_messages'] =
$can_delete_messages;
        }
        if ($can_invite_users !== null)
{
            $args['can_invite_users'] =
$can_invite_users;
        }
        if ($can_restrict_members
!== null) {
```

```
$args['can_restrict_members'] = $can_restrict_members;
        }
        if ($can_pin_messages !== null) {
            $args['can_pin_messages'] = $can_pin_messages;
        }
        if ($can_promote_members !== null) {
            $args['can_promote_members'] = $can_promote_members;
        }
        return $this->Request('promoteChatMember', $args);
    }
    public function setChatPermissions(
        $chat_id,
        array $permissions
    ) {
        $args = [
            'chat_id'     => $chat_id,
            'permissions' => json_encode($permissions)
        ];
        return $this->Request('setChatPermissions', $args);
    }
    public function exportChatInviteLink(
        $chat_id
    ) {
        $args = [
            'chat_id' => $chat_id
        ];
        return $this->Request('exportChatInviteLink', $args);
    }
    public function setChatPhoto(
        $chat_id,
        \CURLFile $photo
    ) {
        $args = [
            'chat_id' => $chat_id,
            'photo'   => $photo
        ];
        return $this->Request('setChatPhoto', $args);
    }
    public function deleteChatPhoto(
        $chat_id
    ) {
        $args = [
```

```
            'chat_id' => $chat_id
        ];
        return $this->Request('deleteChatPhoto', $args);
    }
    public function setChatTitle(
        $chat_id,
        string $title
    ) {
        $args = [
            'chat_id' => $chat_id,
            'title'   => $title
        ];
        return $this->Request('setChatTitle', $args);
    }
    public function setChatDescription(
        $chat_id,
        string $description = null
    ) {
        $args = [
            'chat_id' => $chat_id
        ];
        if ($description !== null) {
            $args['description'] = $description;
        }
        return $this->Request('setChatDescription', $args);
    }
    public function pinChatMessage(
        $chat_id,
        int $message_id,
        bool $disable_notification = null
    ) {
        $args = [
            'chat_id'    => $chat_id,
            'message_id' => $message_id
        ];
        if ($disable_notification !== null) {
            $args['disable_notification'] = $disable_notification;
        }
        return $this->Request('pinChatMessage', $args);
    }
    public function unpinChatMessage(
        $chat_id
    ) {
        $args = [
```

```
            'chat_id' => $chat_id
        ];
        return $this->Request('unpinChatMessage', $args);
    }
    public function leaveChat(
        $chat_id
    ) {
        $args = [
            'chat_id' => $chat_id
        ];
        return $this->Request('leaveChat', $args);
    }
    public function getChat(
        $chat_id
    ) {
        $args = [
            'chat_id' => $chat_id
        ];
        return $this->Request('getChat', $args);
    }
    public function getChatAdministrators(
        $chat_id
    ) {
        $args = [
            'chat_id' => $chat_id
        ];
        return $this->Request('getChatAdministrators', $args);
    }
    public function getChatMembersCount(
        $chat_id
    ) {
        $args = [
            'chat_id' => $chat_id
        ];
        return $this->Request('getChatMembersCount', $args);
    }
    public function getChatMember(
        $chat_id,
        int $user_id
    ) {
        $args = [
            'chat_id' => $chat_id,
            'user_id' => $user_id
        ];
        return $this->Request('getChatMember', $args);
    }
    public function setChatStickerSet(
        $chat_id,
        string $sticker_set_name
```

```
    ) {
        $args = [
            'chat_id'          => $chat_id,
            'sticker_set_name'          =>
$sticker_set_name
        ];
        return                    $this-
>Request('setChatStickerSet',
$args);
    }
    public                    function
deleteChatStickerSet(
        $chat_id
    ) {
        $args = [
            'chat_id' => $chat_id
        ];
        return                    $this-
>Request('deleteChatStickerSet',
$args);
    }
    public                    function
answerCallbackQuery(
        string $callback_query_id,
        string $text = null,
        bool $show_alert = null,
        string $url = null,
        int $cache_time = null
    ) {
        $args = [
            'callback_query_id'          =>
$callback_query_id
        ];
        if ($text !== null) {
            $args['text'] = $text;
        }
        if ($show_alert !== null) {
            $args['show_alert']          =
$show_alert;
        }
        if ($url !== null) {
            $args['url'] = $url;
        }
        if ($cache_time !== null) {
            $args['cache_time']          =
$cache_time;
        }
        return                    $this-
>Request('answerCallbackQuery',
$args);
    }
    public                    function
editMessageText(
        $chat_id = null,
        int $message_id = null,
        string  $inline_message_id  =
null,
        string $text,
        string $parse_mode = null,
```

```
        bool
$disable_web_page_preview      =
null,
        array $reply_markup = null
    ) {
        $args = [
            'text' => $text
        ];
        if ($chat_id !== null) {
            $args['chat_id'] = $chat_id;
        }
        if ($message_id !== null) {
            $args['message_id']        =
$message_id;
        }
        if   ($inline_message_id   !==
null) {
            $args['inline_message_id']
= $inline_message_id;
        }
        if ($parse_mode !== null) {
            $args['parse_mode']        =
$parse_mode;
        }
        if
($disable_web_page_preview   !==
null) {

$args['disable_web_page_preview'
] = $disable_web_page_preview;
        }
        if ($reply_markup !== null) {
            $args['reply_markup']      =
json_encode($reply_markup);
        }
        return                    $this-
>Request('editMessageText',
$args);
    }
    public                    function
editMessageCaption(
        $chat_id = null,
        int $message_id = null,
        string  $inline_message_id  =
null,
        string $caption = null,
        string $parse_mode = null,
        array $reply_markup = null
    ) {
        $args = [];
        if ($chat_id !== null) {
            $args['chat_id'] = $chat_id;
        }
        if ($message_id !== null) {
            $args['message_id']        =
$message_id;
        }
        if   ($inline_message_id   !==
null) {
            $args['inline_message_id']
= $inline_message_id;
        }
```

```
        if ($caption !== null) {
            $args['caption'] = $caption;
        }
        if ($parse_mode !== null) {
            $args['parse_mode'] = $parse_mode;
        }
        if ($reply_markup !== null) {
            $args['reply_markup'] = json_encode($reply_markup);
        }
        return $this->Request('editMessageCaption', $args);
    }
    public function editMessageMedia(
        $chat_id = null,
        int $message_id = null,
        string $inline_message_id = null,
        $media,
        array $reply_markup = null
    ) {
        $args = [
            'media' => $media
        ];
        if ($chat_id !== null) {
            $args['chat_id'] = $chat_id;
        }
        if ($message_id !== null) {
            $args['message_id'] = $message_id;
        }
        if ($inline_message_id !== null) {
            $args['inline_message_id'] = $inline_message_id;
        }
        if ($reply_markup !== null) {
            $args['reply_markup'] = json_encode($reply_markup);
        }
        return $this->Request('editMessageMedia', $args);
    }
    public function editMessageReplyMarkup(
        $chat_id = null,
        int $message_id = null,
        string $inline_message_id = null,
        array $reply_markup = null
    ) {
        $args = [];
        if ($chat_id !== null) {
            $args['chat_id'] = $chat_id;
        }
        if ($message_id !== null) {
            $args['message_id'] = $message_id;
```

```
        }
        if ($inline_message_id !== null) {
            $args['inline_message_id'] = $inline_message_id;
        }
        if ($reply_markup !== null) {
            $args['reply_markup'] = json_encode($reply_markup);
        }
        return $this->Request('editMessageReplyMarkup', $args);
    }
    public function stopPoll(
        $chat_id,
        int $message_id,
        array $reply_markup = null
    ) {
        $args = [
            'chat_id' => $chat_id,
            'message_id' => $message_id
        ];
        if ($reply_markup !== null) {
            $args['reply_markup'] = json_encode($reply_markup);
        }
        return $this->Request('stopPoll', $args);
    }
```

```
    public function deleteMessage(
        $chat_id,
        int $message_id
    ) {
        $args = [
            'chat_id' => $chat_id,
            'message_id' => $message_id
        ];
        return $this->Request('deleteMessage', $args);
    }
    public function sendSticker(
        $chat_id,
        $sticker,
        bool $disable_notification = null,
        int $reply_to_message_id = null,
        array $reply_markup = null
    ) {
        $args = [
            'chat_id' => $chat_id,
            'sticker' => $sticker
        ];
        if ($disable_notification !== null) {

$args['disable_notification'] = $disable_notification;
        }
```

```
        if ($reply_to_message_id !== null) {
            $args['reply_to_message_id'] = $reply_to_message_id;
        }
        if ($reply_markup !== null) {
            $args['reply_markup'] = json_encode($reply_markup);
        }
        return $this->Request('sendSticker', $args);
    }
    public function getStickerSet(
        string $name
    ) {
        $args = [
            'name' => $name
        ];
        return $this->Request('getStickerSet', $args);
    }
    public function uploadStickerFile(
        int $user_id,
        \CURLFile $png_sticker
    ) {
        $args = [
            'user_id'    => $user_id,
            'png_sticker'                => $png_sticker
        ];
        return $this->Request('uploadStickerFile', $args);
    }
    public function createNewStickerSet(
        int $user_id,
        string $name,
        string $title,
        $png_sticker,
        string $emojis,
        bool $contains_masks = null,
        $mask_position = null
    ) {
        $args = [
            'user_id'    => $user_id,
            'name'       => $name,
            'title'      => $title,
            'png_sticker'                => $png_sticker,
            'emojis'     => $emojis
        ];
        if ($contains_masks !== null) {
            $args['contains_masks'] = $contains_masks;
        }
        if ($mask_position !== null) {
            $args['mask_position'] = $mask_position;
```

```
        }
        return $this->Request('createNewStickerSet', $args);
    }
    public function addStickerToSet(
        int $user_id,
        string $name,
        $png_sticker,
        string $emojis,
        $mask_position = null
    ) {
        $args = [
            'user_id'     => $user_id,
            'name'        => $name,
            'png_sticker' => $png_sticker,
            'emojis'      => $emojis
        ];
        if ($mask_position !== null) {
            $args['mask_position'] = $mask_position;
        }
        return $this->Request('addStickerToSet', $args);
    }
    public function setStickerPositionInSet(
        string $sticker,
        int $position
    ) {
        $args = [
            'sticker'  => $sticker,
            'position' => $position
        ];
        return $this->Request('setStickerPositionInSet', $args);
    }
    public function deleteStickerFromSet(
        string $sticker
    ) {
        $args = [
            'sticker' => $sticker
        ];
        return $this->Request('deleteStickerFromSet', $args);
    }
    public function answerInlineQuery(
        string $inline_query_id,
        $results,
        int $cache_time = null,
        bool $is_personal = null,
        string $next_offset = null,
        string $switch_pm_text = null,
```

```
        string $switch_pm_parameter = null
    ) {
        $args = [
            'inline_query_id'  => $inline_query_id,
            'results'          => $results
        ];
        if ($cache_time !== null) {
            $args['cache_time']  = $cache_time;
        }
        if ($is_personal !== null) {
            $args['is_personal']  = $is_personal;
        }
        if ($next_offset !== null) {
            $args['next_offset']  = $next_offset;
        }
        if ($switch_pm_text !== null) {
            $args['switch_pm_text']  = $switch_pm_text;
        }
        if ($switch_pm_parameter !== null) {
            $args['switch_pm_parameter']  = $switch_pm_parameter;
        }
        return $this->Request('answerInlineQuery', $args);
    }
    public function sendInvoice(
        int $chat_id,
        string $title,
        string $description,
        string $payload,
        string $provider_token,
        string $start_parameter,
        string $currency,
        $prices,
        string $provider_data = null,
        string $photo_url = null,
        int $photo_size = null,
        int $photo_width = null,
        int $photo_height = null,
        bool $need_name = null,
        bool $need_phone_number = null,
        bool $need_email = null,
        bool $need_shipping_address = null,
        bool $send_phone_number_to_provider = null,
        bool $send_email_to_provider = null,
        bool $is_flexible = null,
```

```
        bool $disable_notification = null,
        int $reply_to_message_id = null,
        array $reply_markup = null
    ) {
        $args = [
            'chat_id'          => $chat_id,
            'title'            => $title,
            'description'      => $description,
            'payload'          => $payload,
            'provider_token'   => $provider_token,
            'start_parameter'  => $start_parameter,
            'currency'         => $currency,
            'prices'           => $prices
        ];
        if ($provider_data !== null) {
            $args['provider_data']      = $provider_data;
        }
        if ($photo_url !== null) {
            $args['photo_url']          = $photo_url;
        }
        if ($photo_size !== null) {
            $args['photo_size']         = $photo_size;
        }
```

```
        if ($photo_width !== null) {
            $args['photo_width']        = $photo_width;
        }
        if ($photo_height !== null) {
            $args['photo_height']       = $photo_height;
        }
        if ($need_name !== null) {
            $args['need_name']          = $need_name;
        }
        if ($need_phone_number !== null) {
            $args['need_phone_number']  = $need_phone_number;
        }
        if ($need_email !== null) {
            $args['need_email']         = $need_email;
        }
        if ($need_shipping_address !== null) {
            $args['need_shipping_address']  = $need_shipping_address;
        }
        if ($send_phone_number_to_provider !== null) {
```

```
$args['send_phone_number_to_provider'] = $send_phone_number_to_provider;
        }
        if ($send_email_to_provider !== null) {
            $args['send_email_to_provider'] = $send_email_to_provider;
        }
        if ($is_flexible !== null) {
            $args['is_flexible'] = $is_flexible;
        }
        if ($disable_notification !== null) {
            $args['disable_notification'] = $disable_notification;
        }
        if ($reply_to_message_id !== null) {
            $args['reply_to_message_id'] = $reply_to_message_id;
        }
        if ($reply_markup !== null) {
            $args['reply_markup'] = json_encode($reply_markup);
        }
        return $this->Request('sendInvoice', $args);
    }
    public function answerShippingQuery(
        string $shipping_query_id,
        bool $ok,
        $shipping_options = null,
        string $error_message = null
    ) {
        $args = [
            'shipping_query_id' => $shipping_query_id,
            'ok' => $ok
        ];
        if ($shipping_options !== null) {
            $args['shipping_options'] = $shipping_options;
        }
        if ($error_message !== null) {
            $args['error_message'] = $error_message;
        }
        return $this->Request('answerShippingQuery', $args);
    }
    public function answerPreCheckoutQuery(
```

```
        string
$pre_checkout_query_id,
        bool $ok,
        string $error_message = null
    ) {
        $args = [
            'pre_checkout_query_id'
=> $pre_checkout_query_id,
            'ok'                => $ok
        ];
        if ($error_message !== null) {
            $args['error_message']     =
$error_message;
        }
        return                          $this-
>Request('answerPreCheckoutQue
ry', $args);
    }
    public                          function
setPassportDataErrors(
        int $user_id,
        $errors
    ) {
        $args = [
            'user_id' => $user_id,
            'errors'  => $errors
        ];
        return                          $this-
>Request('setPassportDataErrors',
$args);
    }
```

```
    public function sendGame(
        int $chat_id,
        string $game_short_name,
        bool   $disable_notification   =
null,
        int   $reply_to_message_id   =
null,
        array $reply_markup = null
    ) {
        $args = [
            'chat_id'        => $chat_id,
            'game_short_name'        =>
$game_short_name
        ];
        if   ($disable_notification   !==
null) {

$args['disable_notification']        =
$disable_notification;
        }
        if ($reply_to_message_id !==
null) {

$args['reply_to_message_id']      =
$reply_to_message_id;
        }
        if ($reply_markup !== null) {
            $args['reply_markup']      =
json_encode($reply_markup);
        }
```

```
        return $this->Request('sendGame', $args);
    }
    public function setGameScore(
        int $user_id,
        int $score,
        bool $force = null,
        bool $disable_edit_message = null,
        int $chat_id = null,
        int $message_id = null,
        string $inline_message_id = null
    ) {
        $args = [
            'user_id' => $user_id,
            'score'   => $score
        ];
        if ($force !== null) {
            $args['force'] = $force;
        }
        if ($disable_edit_message !== null) {
            $args['disable_edit_message'] = $disable_edit_message;
        }
        if ($chat_id !== null) {
            $args['chat_id'] = $chat_id;
        }
        if ($message_id !== null) {
            $args['message_id'] = $message_id;
        }
        if ($inline_message_id !== null) {
            $args['inline_message_id'] = $inline_message_id;
        }
        return $this->Request('setGameScore', $args);
    }
    public function getGameHighScores(
        int $user_id,
        int $chat_id = null,
        int $message_id = null,
        string $inline_message_id = null
    ) {
        $args = [
            'user_id' => $user_id
        ];
        if ($chat_id !== null) {
            $args['chat_id'] = $chat_id;
        }
        if ($message_id !== null) {
            $args['message_id'] = $message_id;
        }
        if ($inline_message_id !== null) {
```

```
        $args['inline_message_id'] = $inline_message_id;
    }
```

```
    return $this->Request('getGameHighScores', $args);
  }
}
```

3. ApiInterface.php

```
<?php
// namespace TuriBot;
interface ApiInterface
{
  function Request(string $method, array $data);
}
```

4. EasyVars.php

```
<?php
// namespace TuriBot;

class EasyVars
{
  public $type, $chat_type, $chat_id, $message_id, $from_id, $text, $first_name;
  private $types = [
    "message",
    "edited_message",
    "channel_post",
    "edited_channel_post",
  ];
  public function __construct($update)
  {
    foreach ($this->types as $type) {
      if (isset($update->{$type})) {
        $this->type = $type;
        break;
      }
    }
    if (isset($this->type)) {
```

```
        $this->chat_type = $update->{$this->type}->chat->type;
        $this->chat_id = $update->{$this->type}->chat->id;
        $this->message_id = $update->{$this->type}->message_id;
        if (isset($update->{$this->type}->from)) {
            $this->from_id = $update->{$this->type}->from->id;
            $this->first_name = $update->{$this->type}->from->first_name;
        }
        if (isset($update->{$this->type}->text)) {
            $this->text = $update->{$this->type}->text;
        }
    }
}
```

5. Klien.php

```
<?php
// namespace TuriBot;
// use CURLFile;
include 'Api.php';
class Client extends Api
{
    public $easy;
    private $endpoint, $curl, $json_payload;
    /*
     * @param string $token Bot API token
     * @param bool $json_payload if true enable json payload, otherwise use always curl
     */
    public function __construct(string $token, bool $json_payload = false)
    {
        $this->endpoint = "https://api.telegram.org/bot" . $token . "/";
        $this->json_payload = $json_payload;
        $this->curl = curl_init();
        curl_setopt_array($this->curl, [
```

```
CURLOPT_RETURNTRANSFER => true,
        CURLOPT_POST => true,

CURLOPT_FORBID_REUSE => true,
        CURLOPT_HEADER => false,
        CURLOPT_TIMEOUT => 120,

CURLOPT_CONNECTTIMEOUT => 2,

CURLOPT_HTTPHEADER => ["Connection: Keep-Alive", "Keep-Alive: 120"],
      ]);
   }
   /*
    * @return \stdClass of update received from webhook
    */
   public function getUpdate(): \stdClass
   {
      $update = json_decode(file_get_contents("php://input"));
      $this->easy = new EasyVars($update);
      return $update;
   }
   /*
    * @param string $path Path of file
    * @return \CURLFile of $path
    */
   public function inputFile(string $path): \CURLFile
   {
      $path = realpath($path);
      return new CURLFile($path);
   }
   /*
    * Make a request to Bot API
    *
    * @param string $method The method of Bot API
    * @param array $args Argument for the method of Bot API
    *
    * @return \stdClass getUpdate if jsonPayload, otherwise response of Telegram
    */
   public function Request(string $method, array $args = []): \stdClass
```

```
    {
        if ($this->json_payload) {
            $args["method"]             =
$method;
            $request                    =
json_encode($args);
            ob_start();
            header("Content-Type:
application/json");
            header("Connection:
close");
            header("Content-Length: " .
strlen($request));
            echo $request;
            ob_end_flush();
            ob_flush();
            flush();
            $this->json_payload         =
false;
            return $this->getUpdate();
        } else {
            curl_setopt_array($this-
>curl, [
                CURLOPT_URL         =>
$this->endpoint . $method,

CURLOPT_POSTFIELDS          =>
$args,
            ]);
            $resultCurl                 =
curl_exec($this->curl);
            if ($resultCurl === false) {
                $arr = [
                    "ok"            => false,
                    "error_code"            =>
curl_errno($this->curl),
                    "description"           =>
curl_error($this->curl),
                    "curl_error"  => true
                ];
                $resultCurl                 =
json_encode($arr);
            }
            $resultJson                 =
json_decode($resultCurl);
            if ($resultJson === null) {
                $arr = [
                    "ok"            => false,
                    "error_code"            =>
json_last_error(),
                    "description"           =>
json_last_error_msg(),
                    "json_error"  => true
                ];
                $resultJson                 =
json_decode(json_encode($arr));
            }
            return $resultJson;
        }
    }
    /*
```

```
    * Make var_export() and send it in the actual chat_id
    *
    * @param int|string $chat_id for the target chat
    * @param mixed,... $var unlimited optional variable to send
    *
    * @return bool true if can send message, otherwise false
    */
   public function debug($chat_id, ...$vars): bool
   {
      foreach ($vars as $debug) {
         $str = var_export($debug, true);
         $array_str = str_split($str, 4050);
         foreach ($array_str as $value) {
            $result = $this->sendMessage($chat_id, "Debug:" . PHP_EOL . $value);
            if ($result->ok === false) {
               return false;
            }
         }
      }
      return true;
   }
}
```

6. Setlokasi.php

```
<?php

//---------------------- Copyright Block ----------------------
/*

PrayTime.php: Prayer Times Calculator (ver 1.2.2)
Copyright (C) 2007-2010 PrayTimes.org

Developer: Hamid Zarrabi-Zadeh

License: GNU LGPL v3.0

TERMS OF USE:
   Permission is granted to use this code, with or
   without modification, in any website or application
   provided that credit is given to the original work
   with a link back to PrayTimes.org.
```

This program is distributed in the hope that it will
be useful, but WITHOUT ANY WARRANTY.

PLEASE DO NOT REMOVE THIS COPYRIGHT BLOCK.

*/

//---------- Help and Manual ----------
/*

User's Manual:
http://praytimes.org/manual

Calculating Formulas:
http://praytimes.org/calculation

//---------- User Interface ----------

getPrayerTimes (timestamp, latitude, longitude, timeZone)

getDatePrayerTimes (year, month, day, latitude, longitude, timeZone)

setCalcMethod (methodID)
setAsrMethod (methodID)

setFajrAngle (angle)
setMaghribAngle (angle)
setIshaAngle (angle)
setDhuhrMinutes (minutes) // minutes after mid-day
setMaghribMinutes (minutes) // minutes after sunset
setIshaMinutes (minutes) // minutes after maghrib

setHighLatsMethod (methodID) // adjust method for higher latitudes

setTimeFormat (timeFormat)
floatToTime24 (time)
floatToTime12 (time)
floatToTime12NS (time)

//---------- Sample Usage ----------

```
    $prayTime->setCalcMethod($prayTime->ISNA);
    $times = $prayTime->getPrayerTimes(time(), 43, -80, -5);
    print('Sunrise = '. $times[1]);


*/


//---------------------- PrayTime Class ----------------------

class PrayTime
{

    //------------------------ Constants --------------------------


    // Calculation Methods
    var $Jafari    = 0;  // Ithna Ashari
    var $Karachi   = 1;  // University of Islamic Sciences, Karachi
    var $ISNA      = 2;  // Islamic Society of North America (ISNA)
    var $MWL       = 3;  // Muslim World League (MWL)
    var $Makkah    = 4;  // Umm al-Qura, Makkah
    var $Egypt     = 5;  // Egyptian General Authority of Survey
    var $Custom    = 6;  // Custom Setting
    var $Tehran    = 7;  // Institute of Geophysics, University of Tehran


    // Juristic Methods
    var $Shafii    = 0;  // Shafii (standard)
    var $Hanafi    = 1;  // Hanafi


    // Adjusting Methods for Higher Latitudes
    var $None      = 0;  // No adjustment
    var $MidNight  = 1;  // middle of night
    var $OneSeventh = 2;  // 1/7th of night
    var $AngleBased = 3;  // angle/60th of night


    // Time Formats
    var $Time24    = 0;  // 24-hour format
```

```
    var $Time12     = 1;    // 12-hour format
    var $Time12NS   = 2;    // 12-hour format with no suffix
    var $Float      = 3;    // floating point number

    // Time Names
    var $timeNames = array(
        'Fajr',
        'Sunrise',
        'Dhuhr',
        'Asr',
        'Sunset',
        'Maghrib',
        'Isha'
    );

    var $InvalidTime = '-----';       // The string used for invalid times


    //--------------------- Global Variables --------------------


    var $calcMethod    = 0;         // caculation method
    var $asrJuristic   = 0;         // Juristic method for Asr
    var $dhuhrMinutes  = 0;         // minutes after mid-day for Dhuhr
    var $adjustHighLats = 1;        // adjusting method for higher latitudes

    var $timeFormat   = 0;          // time format

    var $lat;         // latitude
    var $lng;         // longitude
    var $timeZone;    // time-zone
    var $JDate;       // Julian date


    //--------------------- Technical Settings --------------------


    var $numIterations = 1;         // number of iterations needed to compute times


    //------------------- Calc Method Parameters --------------------


    var $methodParams = array();
```

```
   /*                                  var
$methodParams[methodNum]     =
array(fa, ms, mv, is, iv);

          fa : fajr angle
          ms : maghrib selector (0 =
angle; 1 = minutes after sunset)
          mv : maghrib parameter
value (in angle or minutes)
          is : isha selector (0 = angle;
1 = minutes after maghrib)
          iv : isha parameter value (in
angle or minutes)
   */


   //-----------------------
Constructors -------------------------


   function   PrayTime($methodID
= 0)
   {

      $this->methodParams[$this-
>Jafari]    = array(16, 0, 4, 0, 14);
      $this->methodParams[$this-
>Karachi]   = array(18, 1, 0, 0, 18);
      $this->methodParams[$this-
>ISNA]      = array(15, 1, 0, 0, 15);
      $this->methodParams[$this-
>MWL]       = array(18, 1, 0, 0, 17);
      $this->methodParams[$this-
>Makkah]    = array(18.5, 1, 0, 1,
90);
      $this->methodParams[$this-
>Egypt]     = array(19.5, 1, 0, 0,
17.5);
      $this->methodParams[$this-
>Tehran]    = array(17.7, 0, 4.5, 0,
14);
      $this->methodParams[$this-
>Custom]    = array(18, 1, 0, 0, 17);

      $this-
>setCalcMethod($methodID);
   }

   function __construct($methodID
= 0)
   {
      $this-
>PrayTime($methodID);
   }



   //--------------------          Interface
Functions --------------------
```

```
    // return prayer times for a given date
    function getDatePrayerTimes($year, $month, $day, $latitude, $longitude, $timeZone)
    {
        $this->lat = $latitude;
        $this->lng = $longitude;
        $this->timeZone = $timeZone;
        $this->JDate = $this->julianDate($year, $month, $day)-$longitude/ (15* 24);
        return $this->computeDayTimes();
    }

    // return prayer times for a given timestamp
    function getPrayerTimes($timestamp, $latitude, $longitude, $timeZone)
    {
        $date = @getdate($timestamp);
        return $this->getDatePrayerTimes($date['year'], $date['mon'], $date['mday'],
                $latitude, $longitude, $timeZone);
    }

    // set the calculation method
    function setCalcMethod($methodID)
    {
        $this->calcMethod = $methodID;
    }

    // set the juristic method for Asr
    function setAsrMethod($methodID)
    {
        if ($methodID < 0 || $methodID > 1)
            return;
        $this->asrJuristic = $methodID;
    }

    // set the angle for calculating Fajr
    function setFajrAngle($angle)
    {
        $this->setCustomParams(array($angle, null, null, null, null));
    }
```

```
   // set the angle for calculating Maghrib
   function setMaghribAngle($angle)
   {
      $this->setCustomParams(array(null, 0, $angle, null, null));
   }

   // set the angle for calculating Isha
   function setIshaAngle($angle)
   {
      $this->setCustomParams(array(null, null, null, 0, $angle));
   }

   // set the minutes after mid-day for calculating Dhuhr
   function setDhuhrMinutes($minutes)
   {
      $this->dhuhrMinutes = $minutes;
   }

   // set the minutes after Sunset for calculating Maghrib
   function setMaghribMinutes($minutes)
   {
      $this->setCustomParams(array(null, 1, $minutes, null, null));
   }

   // set the minutes after Maghrib for calculating Isha
   function setIshaMinutes($minutes)
   {
      $this->setCustomParams(array(null, null, null, 1, $minutes));
   }

   // set custom values for calculation parameters
   function setCustomParams($params)
   {
      for ($i=0; $i<5; $i++)
      {
         if ($params[$i] == null)
            $this->methodParams[$this->Custom][$i] = $this->methodParams[$this->calcMethod][$i];
```

```
            else
                $this->methodParams[$this->Custom][$i] = $params[$i];
        }
        $this->calcMethod = $this->Custom;
    }

    // set adjusting method for higher latitudes
    function setHighLatsMethod($methodID)
    {
        $this->adjustHighLats = $methodID;
    }

    // set the time format
    function setTimeFormat($timeFormat)
    {
        $this->timeFormat = $timeFormat;
    }

    // convert float hours to 24h format
    function floatToTime24($time)
    {
        if (is_nan($time))
            return $this->InvalidTime;
        $time = $this->fixhour($time+ 0.5/ 60);  // add 0.5 minutes to round
        $hours = floor($time);
        $minutes = floor(($time-$hours)* 60);
        return $this->twoDigitsFormat($hours). ':'. $this->twoDigitsFormat($minutes);
    }

    // convert float hours to 12h format
    function floatToTime12($time, $noSuffix = false)
    {
        if (is_nan($time))
            return $this->InvalidTime;
        $time = $this->fixhour($time+ 0.5/ 60);  // add 0.5 minutes to round
        $hours = floor($time);
        $minutes = floor(($time-$hours)* 60);
        $suffix = $hours >= 12 ? ' pm' : ' am';
        $hours = ($hours+ 12- 1)% 12+ 1;
```

```
        return $hours. ':'. $this->twoDigitsFormat($minutes). ($noSuffix ? '' : $suffix);
    }

    // convert float hours to 12h format with no suffix
    function floatToTime12NS($time)
    {
        return $this->floatToTime12($time, true);
    }



    //---------------------- Calculation Functions ----------------------

    // References:
    // http://www.ummah.net/astronomy/saltime
    // http://aa.usno.navy.mil/faq/docs/SunApprox.html


    // compute declination angle of sun and equation of time
    function sunPosition($jd)
    {
        $D = $jd - 2451545.0;
        $g = $this->fixangle(357.529 + 0.98560028* $D);
        $q = $this->fixangle(280.459 + 0.98564736* $D);
        $L = $this->fixangle($q + 1.915* $this->dsin($g) + 0.020* $this->dsin(2*$g));

        $R = 1.00014 - 0.01671* $this->dcos($g) - 0.00014* $this->dcos(2*$g);
        $e = 23.439 - 0.00000036* $D;

        $d = $this->darcsin($this->dsin($e)* $this->dsin($L));
        $RA = $this->darctan2($this->dcos($e)* $this->dsin($L), $this->dcos($L))/ 15;
        $RA = $this->fixhour($RA);
        $EqT = $q/15 - $RA;

        return array($d, $EqT);
    }

    // compute equation of time
    function equationOfTime($jd)
    {
        $sp = $this->sunPosition($jd);
```

```
        return $sp[1];
    }

    // compute declination angle of sun
    function sunDeclination($jd)
    {
        $sp = $this->sunPosition($jd);
        return $sp[0];
    }

    // compute mid-day (Dhuhr, Zawal) time
    function computeMidDay($t)
    {
        $T = $this->equationOfTime($this->JDate+ $t);
        $Z = $this->fixhour(12- $T);
        return $Z;
    }

    // compute time for a given angle G
    function computeTime($G, $t)
    {
        $D = $this->sunDeclination($this->JDate+ $t);
        $Z = $this->computeMidDay($t);
        $V = 1/15* $this->darccos((-$this->dsin($G)- $this->dsin($D)* $this->dsin($this->lat))/
                ($this->dcos($D)* $this->dcos($this->lat)));
        return $Z+ ($G>90 ? -$V : $V);
    }

    // compute the time of Asr
    function computeAsr($step, $t) // Shafii: step=1, Hanafi: step=2
    {
        $D = $this->sunDeclination($this->JDate+ $t);
        $G = -$this->darccot($step+ $this->dtan(abs($this->lat- $D)));
        return $this->computeTime($G, $t);
    }


    //---------------------- Compute Prayer Times ----------------------


    // compute prayer times at given julian date
    function computeTimes($times)
    {
```

```
        $t = $this->dayPortion($times);

        $Fajr = $this->computeTime(180- $this->methodParams[$this->calcMethod][0], $t[0]);
        $Sunrise = $this->computeTime(180- 0.833, $t[1]);
        $Dhuhr = $this->computeMidDay($t[2]);
        $Asr = $this->computeAsr(1+ $this->asrJuristic, $t[3]);
        $Sunset = $this->computeTime(0.833, $t[4]);;
        $Maghrib = $this->computeTime($this->methodParams[$this->calcMethod][2], $t[5]);
        $Isha = $this->computeTime($this->methodParams[$this->calcMethod][4], $t[6]);

        return array($Fajr, $Sunrise, $Dhuhr, $Asr, $Sunset, $Maghrib, $Isha);
    }
```

```
    // compute prayer times at given julian date
    function computeDayTimes()
    {
        $times = array(5, 6, 12, 13, 18, 18, 18); //default times

        for ($i=1; $i<=$this->numIterations; $i++)
            $times = $this->computeTimes($times);

        $times = $this->adjustTimes($times);
        return $this->adjustTimesFormat($times);
    }


    // adjust times in a prayer time array
    function adjustTimes($times)
    {
        for ($i=0; $i<7; $i++)
            $times[$i] += $this->timeZone- $this->lng/ 15;
        $times[2] += $this->dhuhrMinutes/ 60; //Dhuhr
        if ($this->methodParams[$this->calcMethod][1] == 1) // Maghrib
```

```
        $times[5] = $times[4]+ $this->methodParams[$this->calcMethod][2]/ 60;
      if ($this->methodParams[$this->calcMethod][3] == 1) // Isha
        $times[6] = $times[5]+ $this->methodParams[$this->calcMethod][4]/ 60;

      if ($this->adjustHighLats != $this->None)
        $times = $this->adjustHighLatTimes($times);
      return $times;
   }


   // convert times array to given time format
   function adjustTimesFormat($times)
   {
      if ($this->timeFormat == $this->Float)
         return $times;
      for ($i=0; $i<7; $i++)
         if ($this->timeFormat == $this->Time12)
            $times[$i] = $this->floatToTime12($times[$i]);
         else if ($this->timeFormat == $this->Time12NS)
            $times[$i] = $this->floatToTime12($times[$i], true);
         else
            $times[$i] = $this->floatToTime24($times[$i]);
      return $times;
   }


   // adjust Fajr, Isha and Maghrib for locations in higher latitudes
   function adjustHighLatTimes($times)
   {
      $nightTime = $this->timeDiff($times[4], $times[1]); // sunset to sunrise

      // Adjust Fajr
      $FajrDiff = $this->nightPortion($this->methodParams[$this->calcMethod][0])* $nightTime;
      if (is_nan($times[0]) || $this->timeDiff($times[0], $times[1]) > $FajrDiff)
         $times[0] = $times[1]- $FajrDiff;
```

```
        // Adjust Isha
        $IshaAngle = ($this->methodParams[$this->calcMethod][3] == 0) ? $this->methodParams[$this->calcMethod][4] : 18;
        $IshaDiff = $this->nightPortion($IshaAngle)* $nightTime;
        if (is_nan($times[6]) || $this->timeDiff($times[4], $times[6]) > $IshaDiff)
            $times[6] = $times[4]+ $IshaDiff;

        // Adjust Maghrib
        $MaghribAngle = ($this->methodParams[$this->calcMethod][1] == 0) ? $this->methodParams[$this->calcMethod][2] : 4;
        $MaghribDiff = $this->nightPortion($MaghribAngle)* $nightTime;
        if (is_nan($times[5]) || $this->timeDiff($times[4], $times[5]) > $MaghribDiff)
            $times[5] = $times[4]+ $MaghribDiff;

        return $times;
    }


    // the night portion used for adjusting times in higher latitudes
    function nightPortion($angle)
    {
        if ($this->adjustHighLats == $this->AngleBased)
            return 1/60* $angle;
        if ($this->adjustHighLats == $this->MidNight)
            return 1/2;
        if ($this->adjustHighLats == $this->OneSeventh)
            return 1/7;
    }


    // convert hours to day portions
    function dayPortion($times)
    {
        for ($i=0; $i<7; $i++)
            $times[$i] /= 24;
        return $times;
    }



    //---------------------- Misc Functions ----------------------
```

```
   // compute the difference between two times
   function timeDiff($time1, $time2)
   {
      return $this->fixhour($time2-$time1);
   }


   // add a leading 0 if necessary
   function twoDigitsFormat($num)
   {
      return ($num <10) ? '0'. $num : $num;
   }



   //---------------------- Julian Date Functions ----------------------


   // calculate julian date from a calendar date
   function julianDate($year, $month, $day)
   {
      if ($month <= 2)
      {
         $year -= 1;
         $month += 12;
      }
      $A = floor($year/ 100);
      $B = 2- $A+ floor($A/ 4);

      $JD = floor(365.25* ($year+ 4716))+ floor(30.6001* ($month+ 1))+ $day+ $B- 1524.5;
      return $JD;
   }



   // convert a calendar date to julian date (second method)
   function calcJD($year, $month, $day)
   {
      $J1970 = 2440588.0;
      $date = $year. '-'. $month. '-'. $day;
      $ms = strtotime($date);   // # of milliseconds since midnight Jan 1, 1970
      $days = floor($ms/ (1000 * 60 * 60* 24));
      return $J1970+ $days- 0.5;
   }
```

```
    //----------------------
Trigonometric Functions ----------
------------

    // degree sin
    function dsin($d)
    {
        return sin($this->dtr($d));
    }

    // degree cos
    function dcos($d)
    {
        return cos($this->dtr($d));
    }

    // degree tan
    function dtan($d)
    {
        return tan($this->dtr($d));
    }

    // degree arcsin
    function darcsin($x)
    {
        return $this->rtd(asin($x));
    }

    // degree arccos
    function darccos($x)
    {
        return $this->rtd(acos($x));
    }

    // degree arctan
    function darctan($x)
    {
        return $this->rtd(atan($x));
    }

    // degree arctan2
    function darctan2($y, $x)
    {
        return      $this->rtd(atan2($y,
$x));
    }

    // degree arccot
    function darccot($x)
    {
        return $this->rtd(atan(1/$x));
    }

    // degree to radian
    function dtr($d)
    {
        return ($d * M_PI) / 180.0;
    }

    // radian to degree
    function rtd($r)
```

```
    {
        return ($r * 180.0) / M_PI;
    }

    // range reduce angle in degrees.
    function fixangle($a)
    {
        $a = $a - 360.0 * floor($a / 360.0);
        $a = $a < 0 ? $a + 360.0 : $a;
        return $a;
    }

    // range reduce hours to 0..23
    function fixhour($a)
    {
        $a = $a - 24.0 * floor($a / 24.0);
        $a = $a < 0 ? $a + 24.0 : $a;
        return $a;
    }

}

//---------------------- prayTime Object ----------------------

$prayTime = new PrayTime();
?>
```

7. Timezone.php

```
<?php

        // Prayer Times Calculator, Sample Usage
        // By: Hamid Zarrabi-Zadeh
        // Inputs : $method, $year, $latitude, $longitude, $timeZone

        //import_request_variables( "p");
        //extract($_GET, EXTR_PREFIX_ALL, 'p');
        //extract($_POST, EXTR_PREFIX_ALL, 'p');
        include 'PrayTime.php';
```

```
	$latitude=$_GET["latitude"]!=""?$_GET["latitude"]:"24.0465369";
	$longitude = $_GET["longitude"]!=""?$_GET["longitude"]:"120.6842323";
	$timeZone=$_GET["timeZone"]!=""?$_GET["timeZone"]:"+08";
	$year = $_GET["year"]!=""?$_GET["year"]:"2019";
	$method=$_GET["method"]!=""?$_GET["method"]:"3";
/*if (!isset($method) || !isset($year))
		list($method, $year, $latitude, $longitude, $timeZone) = array(0, 2007, 43, -80, -5);
*/
?>
<html>
<head>
	<title>Prayer Timetable</title>
</head>
<style>
	pre {font-family: courier, serif, size: 10pt; margin: 0px 8px;}
</style>
```

```
<body>

<h1> Prayer Timetable </h1>
<form name="form" method="get" action="<?php echo $PHP_SELF ?>">
<div style="padding:10px; background-color: #F8F7F4; border: 1px dashed #EAE9CD;">

	Latitude: <input type="text" value="<?php echo $latitude ?>" name="latitude" size="4">
	Longitude: <input type="text" value="<?php echo $longitude ?>" name="longitude" size="4">
	Time Zone: <input type="text" value="<?php echo $timeZone ?>" name="timeZone" size="2">
	Year: <input type="text" value="<?php echo $year ?>" name="year" size="4"> <br>
	Method:
	<select id="method" name="method" size="1" onchange="document.form.submit()">
```

```
                <option value="0">Shia Ithna-Ashari</option>
                <option value="1">University of Islamic Sciences, Karachi</option>
                <option value="2">Islamic Society of North America (ISNA)</option>
                <option value="3">Muslim World League (MWL)</option>
                <option value="4">Umm al-Qura, Makkah</option>
                <option value="5">Egyptian General Authority of Survey</option>
                <option value="7">Institute of Geophysics, University of Tehran</option>
    </select>
        <input type="submit" value="Make Timetable">

</div>
</form>

<pre>
 Date  Fajr  Sunrise  Dhuhr  Asr  Sunset  Maghrib  Isha
```

```
-----------------------------------------
------------------
<?php

        $prayTime = new PrayTime($method);

        $date = strtotime($year. '-1-1');
        $endDate = strtotime(($year+ 1). '-1-1');

        while ($date < $endDate)
        {
                $times = $prayTime->getPrayerTimes($date, $latitude, $longitude, $timeZone);
                $day = date('M d', $date);
                print $day. "\t". implode("\t", $times). "\n";
                $date += 24* 60* 60;  // next day
        }

?>
</pre>

<script type="text/javascript">
```

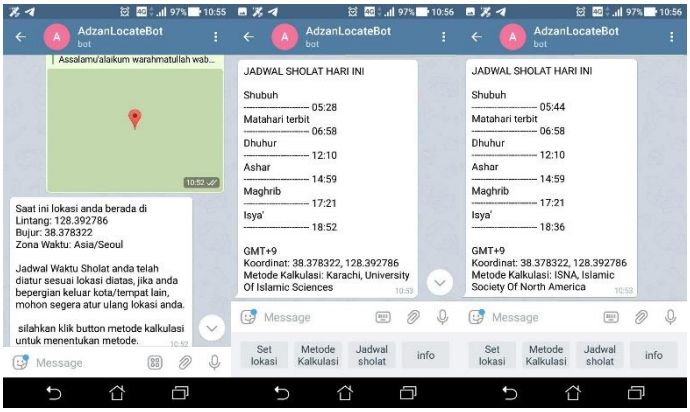
```
        var method = <?php echo $method ?>;
        document.getElementById('method').selectedIndex = Math.min(method, 6);
```

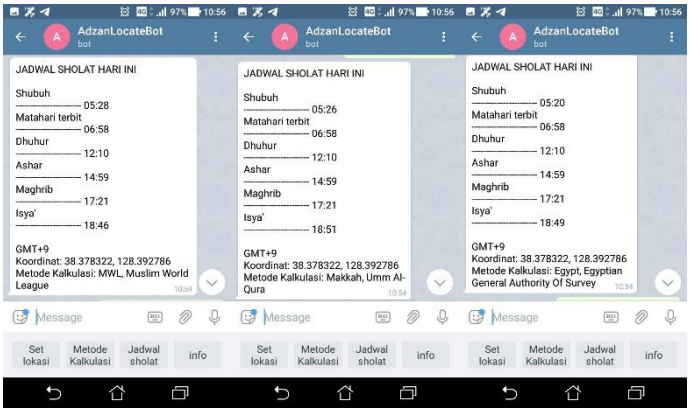
```
</script>

</body>
</html>
```

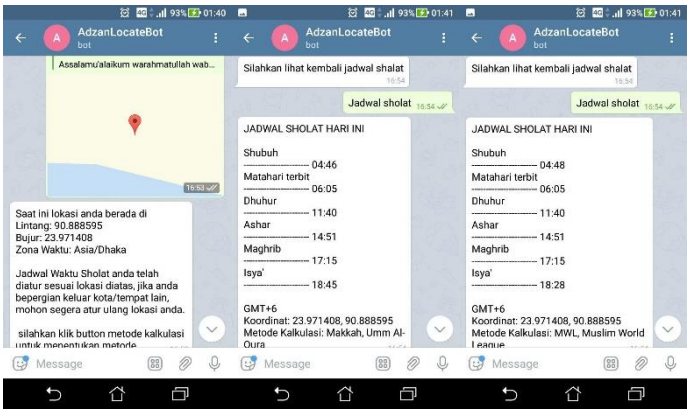
B. Pengujian Fitur

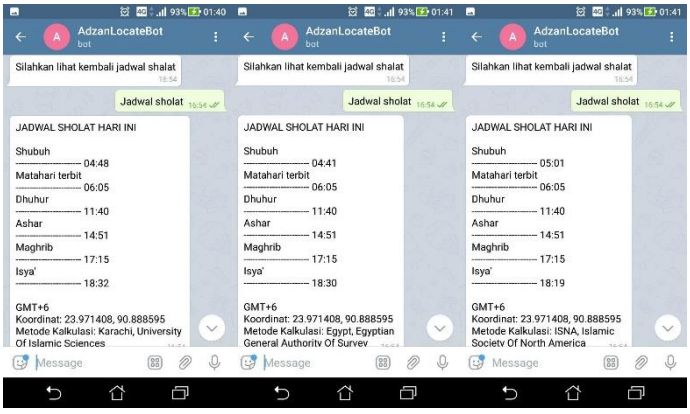
1. Lokasi Korea



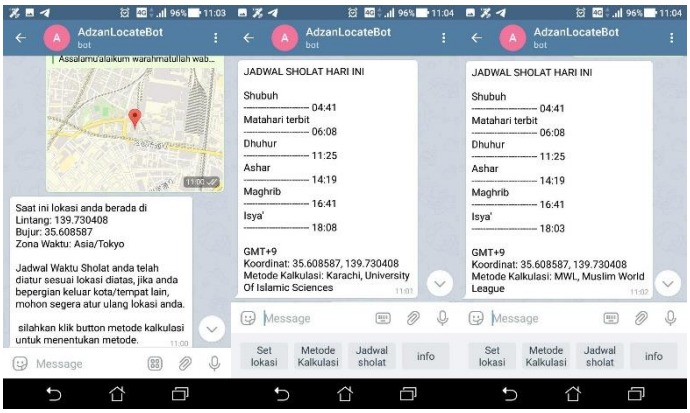


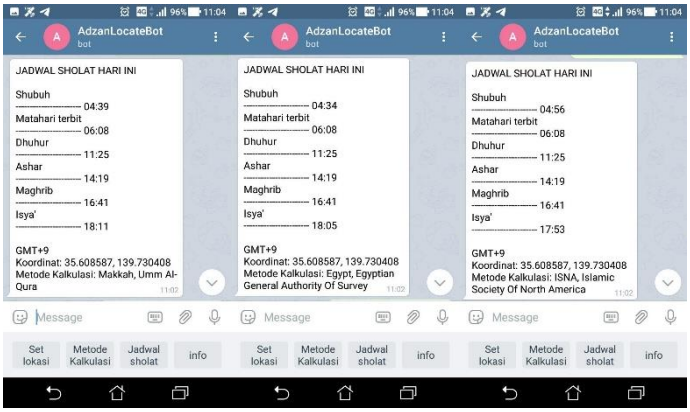
2. Lokasi India



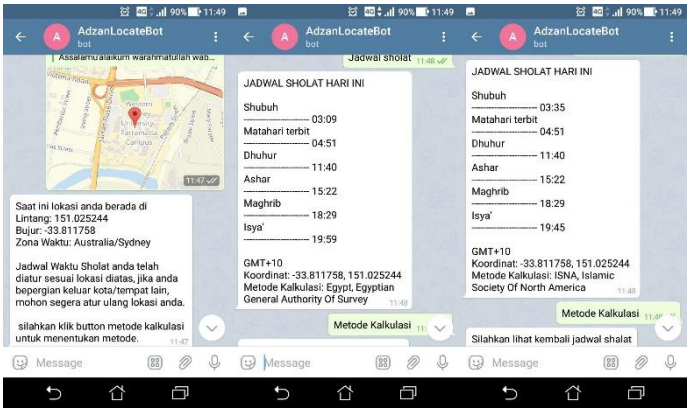


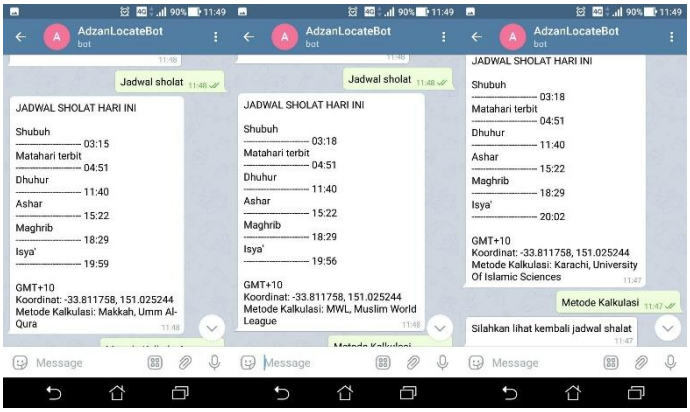
3. Lokasi jepang





4. Lokasi Australia





5. Lokasi Makassar