1. User Requirements

Create an API that can be used to

1. Find out the number of days between two datetime parameters.
2. Find out the number of weekdays between two datetime parameters.
3. Find out the number of complete weeks between two datetime parameters.
4. Accept a third parameter to convert the result of (1, 2 or 3) into one of seconds, minutes, hours, years.
5. Allow the specification of a timezone for comparison of input parameters from different timezones.

2. Questions

1. 1.May I use ISO 8601 format for timezone?
   Yes
2. "1, 2 or 3 into one of seconds, minutes, hours, years. " . Must I provide 12 different parameters? Or may I provide 4 different parameters " seconds, minutes, hours, years", and each query result shows all 3 values?
   The API design is up to you, but the desired response would be the API giving you the difference in only the unit requested. For example if I ask for the difference in seconds, only give me that.
3. Should I take 365 days as one year or must I consider leap years?
   Leap years would be good, but if there are limitations or assumptions in your solution, ensure they're documented

3. Technologies

1. Programming Language and Framework

Choose PHP7 + Laravel as programming language and framework. The security of Laravel is high.

2. Authorise

Laravel Passport, which provides a full OAuth2 server implementation for your Laravel application in a matter of minutes.

3. CORS

Cross-origin resource sharing (CORS) is a mechanism that allows restricted resources on a web page to be requested from another domain outside the domain from which the first resource was served.

If the frontend fetch the API by javascript, I will put Access-Control-Allow-Origin to allow the sample website.

4. Signed Route

ValidateSignature can be used to a signed url for Laravel.

5. HTTPS

When deployment on the server, HTTPS protocol will be used.

6. Parameter Validation

The parameter must not empty. If it is a time with ISO 8601 format, regular expression can be used to check if it is valid. If it is a filter string, an const array can be used to check its validation.

7. Design Pattern

Template method + Strategy + Simple Factory Design Patterns are used in the project.

When validating the parameter, the logic is stable which is first validate the empty then validate the format, so Template Method Design Pattern is a good choice.

When calculating the Date Time Difference, it contains 4 different method. In order to avoid complexity after the algorithm and requirement changes, Strategy Design Pattern is used. If the algorithm changes a lot in the future or a new requirement to calculate the exact float Hours, it will not influence other modules.

Simple Factory Design Pattern is used to create the instance, and with the reflection feature in PHP7, it is easy to create an instance with the filter string.

8. Libraries

Passport and Carbon library in Laravel are used.

Passport library provide Authentication function for the website.

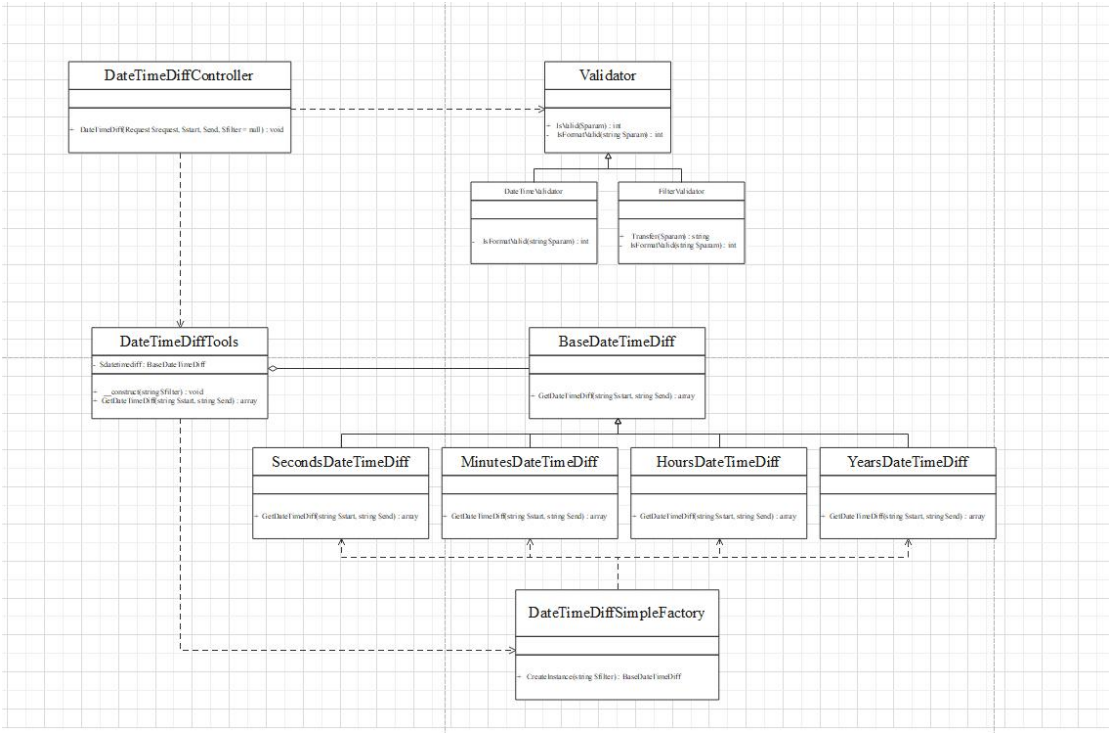
Carbon is a famous date time processing library which provides a lot of functions to deal with the date time in ISO 8601 type. For example, diffInDays function is easy to get the days between 2 date times.

9. Exception

Try-catch is use to capture other exception on the server.

4. Design

1. Class Figure



Validator Classes contain Validator, DateTimeValidator, FilterValidator. The design pattern is Template Method Design Pattern. The class figure is shown below.

Class Name	**DateTimeDiffController**		
Method			
1	DateTimeDiff		
Functionality	API function entry		
	Name	**Type**	**Meaning**
Parameters	$request	Request	Request object
	$start	null or string	Start date time string
	$end	null or string	End date time string
	$filter	null or string	Query filter, default null
Return Value			null

Class Name	Validator		
Method			
1	IsValid		
Functionality	Check if the parameter is valid, public function		
	Name	**Type**	**Meaning**
Parameters	$param	null or string	a parameter to be checked
Return Value		int	0: valid 1: empty

			2: format error
2	IsFormatValid		
Functionality	Check if the parameter format is valid, abstract protected function		
	Name	**Type**	**Meaning**
Parameters	$param	string	a parameter to be checked
Return Value		int	0: valid 2: format error

Class Name	DateTimeValidator		
Method			
1	IsFormatValid		
Functionality	Check if the parameter format is valid, abstract protected function		
	Name	**Type**	**Meaning**
Parameters	$param	string	a parameter to be checked
Return Value		int	0: valid 2: format error

Class Name	FilterValidator		
Method			
1	Transfer		
Functionality	Transfer empty to ‘Base’ and transfer ’Base’ to ‘Other’ in Order to use reflection		
	Name	**Type**	**Meaning**
Parameters	$param	null or string	a parameter to be transfer
Return Value		string	‘Base’: days query ‘Other’: invalid query
2	IsFormatValid		
Functionality	Check if the parameter format is valid, abstract protected function		
	Name	**Type**	**Meaning**
Parameters	$param	string	a parameter to be checked
Return Value		int	0: valid 2: format error

Class Name	DateTimeDiffSimpleFactory		
Method			
1	CreateInstance		
Functionality	Create an object base on the parameter string, public function		
	Name	**Type**	**Meaning**
Parameters	$filter	string	A type of instance
Return Value		BaseDateTimeDiff	an object: object of BaseDateTimeDiff

Class Name	DateTimeDiffTools		
Property			
1	datetimediff		
Meaning	An instance of BaseDateTimeDiff, private property		
Method			
1	__construct		
Functionality	Constructor, public function		
	Name	**Type**	**Meaning**
Parameters	$filter	string	Decide the type of DateTimeDiff instance
Return Value			Null
2	GetDateTimeDiff		
Functionality	Get the difference of 2 date times, public function		
	Name	**Type**	**Meaning**
Parameters	$start	string	Start date time string
	$end	string	End date time string
Return Value		array	An array stores the number of days, weekdays and complete weeks

Class Name	BaseDateTimeDiff		
Method			
1	GetDateTimeDiff		
Functionality	Get the difference of 2 date times, public function		
	Name	**Type**	**Meaning**
Parameters	$start	string	Start date time string
	$end	string	End date time string
Return Value		array	An array stores the number of days, weekdays and complete weeks

Class Name	SecondsDateTimeDiff		
Method			
1	GetDateTimeDiff		
Functionality	Get the difference of 2 date times, public function		
	Name	**Type**	**Meaning**
Parameters	$start	string	Start date time string
	$end	string	End date time string
Return Value		array	An array stores the number of days, weekdays and complete weeks

Class Name	MinutesDateTimeDiff		
Method			
1	GetDateTimeDiff		
Functionality	Get the difference of 2 date times, public function		
	Name	**Type**	**Meaning**
Parameters	$start	string	Start date time string
	$end	string	End date time string
Return Value		array	An array stores the number of days, weekdays and complete weeks

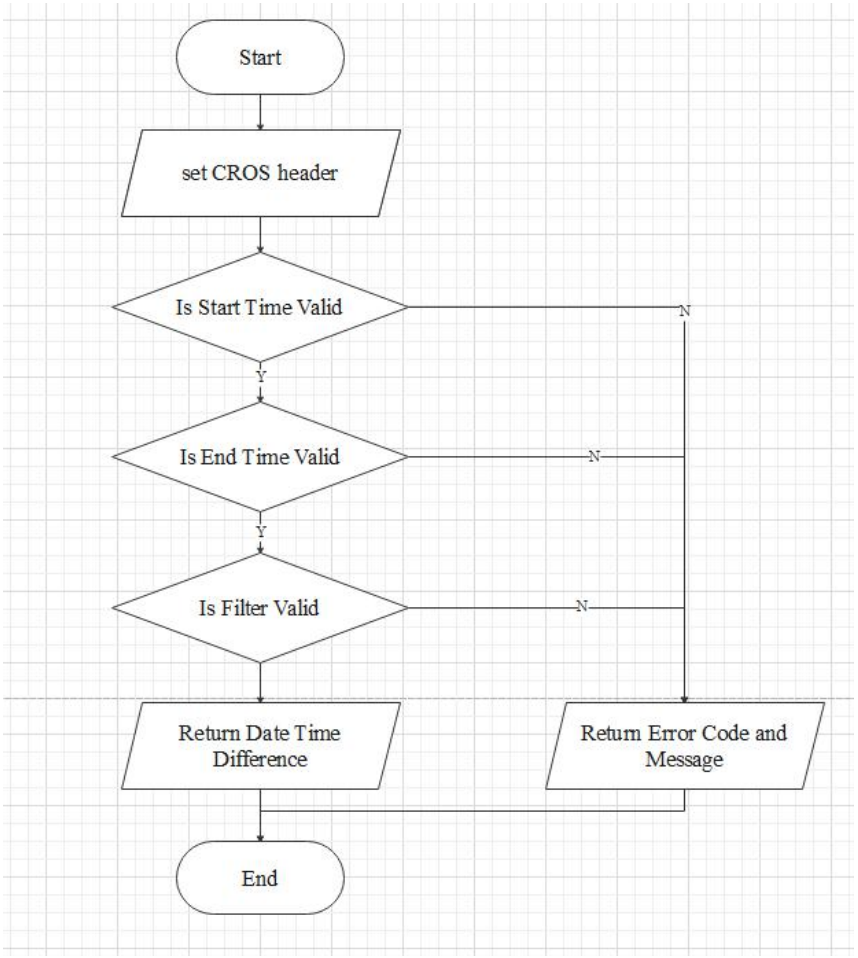
Class Name	HoursDateTimeDiff		
Method			
1	GetDateTimeDiff		
Functionality	Get the difference of 2 date times, public function		
	Name	**Type**	**Meaning**
Parameters	$start	string	Start date time string
	$end	string	End date time string
Return Value		array	An array stores the number of days, weekdays and complete weeks

Class Name	YearsDateTimeDiff		
Method			
1	GetDateTimeDiff		
Functionality	Get the difference of 2 date times, public function		
	Name	**Type**	**Meaning**
Parameters	$start	string	Start date time string
	$end	string	End date time string
Return Value		array	An array stores the number of days, weekdays and complete weeks

2. Endpoint

Create 4 Endpoints for API, 2 for Public and 2 for Authentication. Each has one with filter and one without filter.

3. API Work flow Figure



4. API Return Value Design

The return value of API contains code, state, message and data.

code	200	Request succeed
	40011	Empty string error
	40012	Format error
	500	Server error
state	success	Request succeed
	error	Request failed
message	1	The start datetime is empty.
	2	The end datetime is empty.
	3	The filter is empty.
	4	The start datetime format is not ISO 8601 format.
	5	The end datetime format is not ISO 8601 format.
	6	The filter format is wrong.
	7	Date time diff query succeed.
	8	Other exceptional message
data	1	$data[‘days’] $data[‘weeks’] $data[‘complete_weeks’]
	2	$data[‘days_seconds’] $data[‘weeks_seconds’]

		$data[‘complete_weeks_seconds’]
	3	$data[‘days_minutes’] $data[‘weeks_minutes’] $data[‘complete_weeks_minutes’]
	4	$data[‘days_hours’] $data[‘weeks_hours’] $data[‘complete_weeks_hours’]
	5	$data[‘days_years’] $data[‘weeks_years’] $data[‘complete_weeks_years’]

5. Code

1. Create Laravel Project DateTimeDiffService

```
composer create-project laravel/laravel blog --prefer-dist
```

2. Configure nesbot/carbon and laravel/passport

```
composer require nesbot/carbon
composer require laravel/passport
```

3. Create Class Structure, Test Classes, Controller and Endpoint

   Manually create DateTimeDiff Folder and Classes:

   - DateTimeDiffTool.php
   - BaseDateTimeDiff.php
   - SecondsDateTimeDiff.php
   - MinutesDateTimeDiff.php
   - HoursDateTimeDiff.php
   - YearsDateTimeDiff.php

   SimpleFactory Folder and Classes:

   - DateTimeDiffSimpleFactory.php

   Validator Folder and Classes:

   - Validator.php
   - DateTimeValidator.php
   - FilterValidator.php

   Use php artisan create Test Classes.

```
php artisan make:test DateTimeDiffTest
php artisan make:test BaseDateTimeDiffTest --unit
php artisan make:test SecondsDateTimeDiffTest --unit
php artisan make:test MinutesDateTimeDiffTest --unit
php artisan make:test HoursDateTimeDiffTest --unit
php artisan make:test YearsDateTimeDiffTest --unit
php artisan make:test DateTimeDiffSimpleFactoryTest --unit
php artisan make:test DateTimeValidatorTest --unit
php artisan make:test FilterValidatorTest --unit
```

   Use php artisan create DateTimeDiffController which provides a function DateTimeDiff

```
php artisan make:controller Api/v1/DateTimeDiffController
public function DateTimeDiff(Request $request, $start, $end, $filter = null)
```

   Create Endpoint

```
Route::group(['prefix' => 'v1'], function ()
{
    Route::get('/datetimediff/{start}/{end}', 'Api\v1\DateTimeDiffController@DateTimeDiff');
    Route::get('/datetimediff/{start}/{end}/{filter}', 'Api\v1\DateTimeDiffController@DateTimeDiff');
});
```

   Configure composer.json import Classes

```
"autoload": {
```

```
    "psr-4": {
        "App\\": "app/"
    },
        "classmap": [
        "database/seeds",
        "database/factories",
        "app/Libraries/Classes/DateTimeDiff",
        "app/Libraries/Classes/SimpleFactory",
        "app/Libraries/Classes/Validator"
    ]
}
```

Execute composer command

```
composer dump-autoload
```

4. Implement Validator Classes and do the Unit Test

Validator Classes use Template Method Design Pattern. Validator Class is base class that contains a function IsValid($param) which is a stable template to execute string empty validation and format validation. DatetimeValidator Class extends Validator Class and use regular expression function to overrides format validation. FilterValidator Class extends Validator Class and use array to check if the filter format is right or not. It also provides a public function Transfer($param) to deal with the BaseDateTimeDiff query.

5. Implement DateTimeDiffSimpleFactory Classes and do the Unit Test

DateTimeDiffSimpleFactory Classes use Simple Factory Design Pattern. DateTimeDiffSimpleFactory Class provides function CreateInstance(string $filter) which executes PHP Reflection to create a new instance of children objects. The reflection must use full path string instead of use namespace above the file.

6. Implement BaseDateTimeDiff Classes and do the Unit Test

BaseDateTimeDiff Class provides function GetDateTimeDiff(string $start, string $end) which returns an array containing the query results. Carbon Methods diffInDays, diffInWeekdays and diffinweeks are easy to use, but when testing, an obvious error happens like below.

```
$diff=$datetimediff->GetDateTimeDiff('2020-07-10T23:00:00+09:30', '2020-07-10T23:00:01+09:30');
$this->assertEquals($diff, ['days' => 0, 'weekdays' => 1, 'compete_weeks' => 0]);
```

The days difference is 0, but the weekdays difference is 1, which is not logical. So I test all the situations of different dates and I found first of all, weekdays depend on local time and only consider the date without time. Then If the end day is a weekday and the time of the start day is greater or equal to that of the end day, the result must be added 1 day. Finally minus 1 day which not a whole day of 24 hours. So I designed an algorithm to modify the function diffInWeekdays to make it correct. After that, I test all the situations.

7. Implement DateTimeDiff Classes and do the Unit Test

According to the requirement converting the result of days to seconds, minutes, hours and years. So the logic is very easy. But Strategy Design Pattern is applied, which makes it easy to change if the requirement changes.

8. Implement DateTimeDiffTools Classes

DateTimeDiffTools Class is the Client of Strategy Design Pattern, which transfers the filter name, using factory to create an object of BaseDateTimeDiff Class. The public function

GetDateTimeDiff(string $start, string $end) is provided to the DateTimeDiffController.

9. Integration and Feature Test
   Implement the main logic of API and Do the Feature Test

6. Unit Test and Feature Test

1. Validator Unit Test

DateTimeValidatorTest covers empty and all kinds of ISO8601 or related error strings. Totally 41 cases.(Just list a group of test cases for example)

Test case	Return Value	Description
2020-01-01	0	Standard year, month, day
2020-1-01	0	1 digit month
2020-01-1	0	1 digit day
2020-1-1	0	1 digit month, 1 digit day
202-01-01	2	3 digits year
20200-01-01	2	5 digits year
2020-01-001	2	3 digits day
2020-001-01	2	3 digits month
2020-001-001	2	3 digits month, 3 digits day
2020-13-01	2	Ones place of month is wrong
2020-21-01	2	Tens place of month is wrong
2020-01-41	2	Tens place of day is wrong

FilterValidatorTest covers empty and all kinds of uppercase and lowercase letters or spelling errors. Totally 41 cases.

2. DateTimeDiffSimpleFactory Unit Test

DateTimeDiffSimpleFactoryTest covers all 5 DateTimeDiff Class Instance Generator. Totally 5 cases.

3. BaseDateTimeDiff Unit Test

BaseDateTimeDiffTest covers all the cases of different times.Totally 37 cases.

4. DateTimeDiff Unit Test

SecondsDateTimeDiffTest, MinutesDateTimeDiffTest, HoursDateTimeDiffTest, YearsDateTimeDiffTest cover the cases of different times. Totally 119 cases.

5. Feature Test

DateTimeDiffTest is Feature Test, and it covers the cases of different request. Totally 26 cases.

7. Deployment

DateTimeDiffService provides an API to get the difference between two Date Times. The project deployment on CentOS7 steps are below:

1. Update Package Repository Cache

   Before you start building the stack, be sure to update the packages on your CentOS 7 server using the command:

   ```
   sudo yum update
   ```

2. Install the Apache Web Server

   Install Apache on Centos with

   ```
   sudo yum install httpd
   ```

   Finally, set up Apache to start at boot

   ```
   sudo systemctl enable httpd.service
   ```

3. Install MySQL (MariaDB) and Create a Database

   Install MariaDB with the command

   ```
   sudo yum install mariadb-server mariadb
   ```

   Now start MariaDB using the command

   ```
   sudo systemctl start mariadb
   ```

4. Run MySQL Security Script

   Begin by typing the command

   ```
   sudo mysql_secure_installation
   ```

   Lastly, enable MariaDB to start up when you boot the system

   ```
   sudo systemctl enable mariadb.service
   ```

5. Install PHP

   Install the MySQL extension along with PHP, again using the yum package installer, with the command

   ```
   yum install epel-release
   yum install http://rpms.remirepo.net/enterprise/remi-release-7.rpm
   yum install -y php73-php-fpm php73-php-cli php73-php-bcmath php73-php-gd php73-php-json php73-php-mbstring php73-php-mcrypt php73-php-mysqlnd php73-php-opcache php73-php-pdo php73-php-pecl-crypto php73-php-pecl-mcrypt php73-php-pecl-geoip php73-php-recode php73-php-snmp php73-php-soap php73-php-xmll
   systemctl enable php73-php-fpm
   systemctl start php73-php-fpm
   yum-config-manager --enable remi-php73
   yum -y install php php-opcache
   yum install php-mbstring
   yum install php-dom
   ```

   To have your Apache webserver start co-working with PHP, restart the server

   ```
   sudo systemctl restart httpd.service
   ```

6. Install Git

   ```
   yum install git
   ```

7. Install Composer

```
cd /tmp
sudo curl -sS https://getcomposer.org/installer | php
mv composer.phar /usr/local/bin/composer
```

8. Enter into Apache www folder, git clone the project DateTimeDiffService

```
git clone https://github.com/fk827728/DateTimeDiffService.git
```

9. Enter MySQL, source datetimediff.sql in documents folder of project folder
   This is the database used to Authentication

```
mysql -uroot -p
>> source /var/www/html/DateTimeDiffService/documents/datetimediff.sql;
```

10. Enter into the folder DateTimeDiffService, composer install

```
composer install
```

11. Copy .env.example to .env and edit and save DB_USERNAME and DB_PASSWORD

```
cp .env.example .env
vi .env
```

12. Make the fold storage permission

```
chmod -R 777 storage
```

13. The installation is finished and Chrome Browser or Postman can be used to test the url below:
    Authenticate URL
    http://server_ip/DateTimeDiffService/public/api/v1/datetimediff/2020-01-01/2020-01-02
    Public URL
    http://server_ip/DateTimeDiffService/public/api/v1/public/datetimediff/2020-01-01/2020-01-02