***LOGS***

LIKES.LOG

2022-01-16 21:40:36 info: Logging: userid=10, LikesController like() was successful, ip=::1, tradeid=76

2022-01-16 21:40:51 info: Logging: userid=10, LikesController dislike() was successful, ip=::1, tradeid=76

TRADES.LOG

2022-01-16 21:31:46 info: Logging: userid=10, InvestmentController add() was successful, ip=::1, tradeid=102

2022-01-16 21:33:46 info: Logging: userid=10, InvestmentController add() was successful, ip=::1, tradeid=103

2022-01-16 21:55:24 info: Logging: userid=11, InvestmentController add() was successful, ip=::1, tradeid=104

***CONTROLLERS***

API CONTROLLER

<?php

namespace App\Controller;

class ApiController extends AppController{

    //authentication

   //reference: https://book.cakephp.org/4/en/tutorials-and-examples/cms/authentication.html#adding-password-hashing

   public function beforeFilter(\Cake\Event\EventInterface $event)

   {

      parent::beforeFilter($event);

      // Configure the login action to not require authentication, preventing

      // the infinite redirect loop issue

      $this->Authentication->addUnauthenticatedActions(['index','view','delete','add']);

         //allow unauthenicated users to access pages inside parameter -->login

   }

   //Get list of trade and the like counter

   public function view($id){//http://localhost:8084/assignment/greek\_investment\_app\_not\_working/api/80.json

    $investmentsTable = $this->fetchTable('Investments');

        $likesTable = $this->fetchTable('likes');

        //$allInvestments = $investmentsTable->findById($id);

        $allInvestments = $investmentsTable->findById($id)->contain(['Tickers'])->all();

        //pr($allInvestments);

        $this->set('allInvestments',$allInvestments);

        $this->viewBuilder()->setOption('serialize', ['allInvestments']);

    }

    public function add()//api.json

    {

        $this->request->allowMethod(['post', 'put']);

        $recipe = $this->request->getData();

        $selectPrivacy = array('True','False');

        //send to add

        $this->set('selectTicker',$selectTicker);//send to view

        $this->set('selectPrivacy',$selectPrivacy);//send to view

        if ($this->request->is("post")) {

            $investmentsTable = $this->fetchTable('Investments');

            $data['share'] = strip\_tags($this->request->getData('share'));

            $data['bought\_at'] = strip\_tags($this->request->getData('bought\_at'));

            $newInvetsment = $investmentsTable->newEntity($data);

            //get loggedinuserid

            $user = $this->loggedInUser->get('id');//get user id from controller

            $newInvetsment->set('user\_id', $user);

            if($investmentsTable->save($newInvetsment)){//push data

                $message = 'Saved';

            }else{

                $message = 'Error';

            }

        }

        $this->set([

            'message' => $message,

            'recipe' => $newInvetsment,

        ]);

        $this->viewBuilder()->setOption('serialize', ['recipe', 'message']);

    }

   public function delete($id){///api/80.json

        $this->request->allowMethod(['delete']);

        $investmentsTable = $this->fetchTable('Investments');

        $investmentToDelete = $investmentsTable->get($id);

        $likesTable = $this->fetchTable('Likes');

        $likesToDelete = $likesTable->find()->where(['investment\_id'== $id])->first();

        $likesExists = $likesTable->exists(['investment\_id' => $id]);

        $message = '';

        if($likesExists){//delete likes table before deleting the investments table due to errors

            $likesTable->delete($likesToDelete);

            if($investmentsTable->delete($investmentToDelete)){

                $message = "Investment and likes has been removed!";

            }else{

                $message = "Could not remove likes and investment!";

            }

        }else{//else in no likes exist

            if($investmentsTable->delete($investmentToDelete)){

                $message = 'Deleted';"Investment has been removed!";

            }else{

                $message = "Could not remove investment!";

            }

        }

        $this->set('message', $message);

        $this->viewBuilder()->setOption('serialize', ['message']);

   }

}

APP CONTROLLER

<?php

namespace App\Controller;

class ApiController extends AppController{

    //authentication

   //reference: https://book.cakephp.org/4/en/tutorials-and-examples/cms/authentication.html#adding-password-hashing

   public function beforeFilter(\Cake\Event\EventInterface $event)

   {

      parent::beforeFilter($event);

      // Configure the login action to not require authentication, preventing

      // the infinite redirect loop issue

      $this->Authentication->addUnauthenticatedActions(['index','view','delete','add']);

         //allow unauthenicated users to access pages inside parameter -->login

   }

   //Get list of trade and the like counter

   public function view($id){//http://localhost:8084/assignment/greek\_investment\_app\_not\_working/api/80.json

    $investmentsTable = $this->fetchTable('Investments');

        $likesTable = $this->fetchTable('likes');

        //$allInvestments = $investmentsTable->findById($id);

        $allInvestments = $investmentsTable->findById($id)->contain(['Tickers'])->all();

        //pr($allInvestments);

        $this->set('allInvestments',$allInvestments);

        $this->viewBuilder()->setOption('serialize', ['allInvestments']);

    }

    public function add()//api.json

    {

        $this->request->allowMethod(['post', 'put']);

        $recipe = $this->request->getData();

        $selectPrivacy = array('True','False');

        //send to add

        $this->set('selectTicker',$selectTicker);//send to view

        $this->set('selectPrivacy',$selectPrivacy);//send to view

        if ($this->request->is("post")) {

            $investmentsTable = $this->fetchTable('Investments');

            $data['share'] = strip\_tags($this->request->getData('share'));

            $data['bought\_at'] = strip\_tags($this->request->getData('bought\_at'));

            $newInvetsment = $investmentsTable->newEntity($data);

            //get loggedinuserid

            $user = $this->loggedInUser->get('id');//get user id from controller

            $newInvetsment->set('user\_id', $user);

            if($investmentsTable->save($newInvetsment)){//push data

                $message = 'Saved';

            }else{

                $message = 'Error';

            }

        }

        $this->set([

            'message' => $message,

            'recipe' => $newInvetsment,

        ]);

        $this->viewBuilder()->setOption('serialize', ['recipe', 'message']);

    }

   public function delete($id){///api/80.json

        $this->request->allowMethod(['delete']);

        $investmentsTable = $this->fetchTable('Investments');

        $investmentToDelete = $investmentsTable->get($id);

        $likesTable = $this->fetchTable('Likes');

        $likesToDelete = $likesTable->find()->where(['investment\_id'== $id])->first();

        $likesExists = $likesTable->exists(['investment\_id' => $id]);

        $message = '';

        if($likesExists){//delete likes table before deleting the investments table due to errors

            $likesTable->delete($likesToDelete);

            if($investmentsTable->delete($investmentToDelete)){

                $message = "Investment and likes has been removed!";

            }else{

                $message = "Could not remove likes and investment!";

            }

        }else{//else in no likes exist

            if($investmentsTable->delete($investmentToDelete)){

                $message = 'Deleted';"Investment has been removed!";

            }else{

                $message = "Could not remove investment!";

            }

        }

        $this->set('message', $message);

        $this->viewBuilder()->setOption('serialize', ['message']);

   }

}

APP CONTROLLER

<?php

declare(strict\_types=1);

/\*\*

 \* CakePHP(tm) : Rapid Development Framework (https://cakephp.org)

 \* Copyright (c) Cake Software Foundation, Inc. (https://cakefoundation.org)

 \*

 \* Licensed under The MIT License

 \* For full copyright and license information, please see the LICENSE.txt

 \* Redistributions of files must retain the above copyright notice.

 \*

 \* @copyright Copyright (c) Cake Software Foundation, Inc. (https://cakefoundation.org)

 \* @link      https://cakephp.org CakePHP(tm) Project

 \* @since     0.2.9

 \* @license   https://opensource.org/licenses/mit-license.php MIT License

 \*/

namespace App\Controller;

use Cake\Controller\Controller;

/\*\*

 \* Application Controller

 \*

 \* Add your application-wide methods in the class below, your controllers

 \* will inherit them.

 \*

 \* @link https://book.cakephp.org/4/en/controllers.html#the-app-controller

 \*/

class AppController extends Controller

{

    /\*\*

     \* Initialization hook method.

     \*

     \* Use this method to add common initialization code like loading components.

     \*

     \* e.g. `$this->loadComponent('FormProtection');`

     \*

     \* @return void

     \*/

    public $loggedInUser;

    public function initialize(): void

    {

        parent::initialize();

        $this->loadComponent('RequestHandler');

        $this->loadComponent('Flash');

        $this->loadComponent('Authentication.Authentication');//loads the plugin

        //if user is logged in...

        if($user = $this->Authentication->getIdentity()){

            //send the logged in user as $loggedInUser to ALL views (because we are in hte app controller)

            $this->set('loggedInUser',$user);//VISABLE FROM ALL THE VIEWS

            $this->loggedInUser = $user;//VISABLE FROM ALL THE CONTROLLERS (PUBLIC)

        }

        /\*

         \* Enable the following component for recommended CakePHP form protection settings.

         \* see https://book.cakephp.org/4/en/controllers/components/form-protection.html

         \*/

        //$this->loadComponent('FormProtection');

    }

    public function beforeFilter(\Cake\Event\EventInterface $event)

    {

        parent::beforeFilter($event);

        // Configure the login action to not require authentication, preventing

        // the infinite redirect loop issue

        $this->Authentication->addUnauthenticatedActions(['login']);

    }

}

INVESTMENTS CONTROLLER

<?php

namespace App\Controller;

use Cake\Log\Log;

use Cake\Log\Engine\FileLog;

use Cake\ORM\Locator\LocatorAwareTrait;

class InvestmentsController extends AppController{

   public function index(){

        $investmentsTable = $this->fetchTable('Investments');

        $likesTable = $this->fetchTable('likes');

        $allInvestments = $investmentsTable->find()->contain(['Tickers'])->all();

        $allLikes = $likesTable->find('all');

        $user = $this->loggedInUser->get('email');//get user id from controller

        $session = $this->loggedInUser->get('id');

        //$first\_name = $this->loggedInUser->get('first\_name');

        $last\_name = $this->loggedInUser->get('last\_name');

        $usersTable = $this->fetchTable('Users');

        $this->set('usersTable', $usersTable);

        $this->set('user\_username', $user);

        $this->set('user\_session', $session);

        $this->set('allInvestments', $allInvestments);

        $this->set('likesTable', $likesTable);

        $this->set('allLikes', $allLikes);

   }

    public function add(){

        $data = $this->request->getData();//get POST data

        //retrieve tickers and list items into select for options

        $tickersTable = $this->fetchTable('Tickers');

        $query = $tickersTable->find('list');

        $selectTicker = $query->toArray();

        //select privacy

        $selectPrivacy = array('True','False');

        //send to add

        $this->set('selectTicker',$selectTicker);//send to view

        $this->set('selectPrivacy',$selectPrivacy);//send to view

        if ($this->request->is("post")) {

            $investmentsTable = $this->fetchTable('Investments');

            $data['share'] = strip\_tags($this->request->getData('share'));

            $data['bought\_at'] = strip\_tags($this->request->getData('bought\_at'));

            $newInvetsment = $investmentsTable->newEntity($data);

            //get loggedinuserid

            $user = $this->loggedInUser->get('id');//get user id from controller

            $newInvetsment->set('user\_id', $user);

            if($investmentsTable->save($newInvetsment)){//push data

                $this->Flash->success("Investment added!");

                Log::setConfig('trades', [

                    'className' => FileLog::class,

                    'path' => LOGS,

                    'levels' => ['info'],

                    'scopes' => ['trades'],

                    'file' => 'trades.log',

                ]);

                Log::info('Logging: userid={user}, {message}, ip={ipaddress}, tradeid={trade\_id}', ['scope' => ['trades'], 'user' => $this->loggedInUser->get('id'),

                'ipaddress' => $this->request->clientIp(),

                'message'=> 'InvestmentController add() was successful',

                'trade\_id'=> $newInvetsment->id]);

            }else{

                $this->Flash->error("Error: InvestmentsController -- Investment not added");

                Log::setConfig('trades', [

                    'className' => FileLog::class,

                    'path' => LOGS,

                    'levels' => ['error'],

                    'scopes' => ['trades'],

                    'file' => 'trades.log',

                ]);

                Log::error('Logging: userid={user}, {message}, ip={ipaddress}, tradeid={trade\_id}', ['scope' => ['trades'], 'user' => $this->loggedInUser->get('id'),

                'ipaddress' => $this->request->clientIp(),

                'message'=> 'InvestmentController add() was unsuccessful',

                'trade\_id'=> $newInvetsment->id]);

            }

        }

    }

    public function delete($id){//id is recieved from index

        //reference: https://book.cakephp.org/4/en/orm/deleting-data.html

        $investmentsTable = $this->fetchTable('Investments');

        $investmentToDelete = $investmentsTable->get($id);

        $likesTable = $this->fetchTable('Likes');

        $likesToDelete = $likesTable->find()->where(['investment\_id'== $id])->first();

        $likesExists = $likesTable->exists(['investment\_id' => $id]);

        if($likesExists){//delete likes table before deleting the investments table due to errors

            $likesTable->delete($likesToDelete);

            if($investmentsTable->delete($investmentToDelete)){

                $this->Flash->success("Investment has been removed!");

            }else{

                $this->Flash->error("Could not remove investment!");

            }

        }else{

            if($investmentsTable->delete($investmentToDelete)){

                $this->Flash->success("Investment has been removed!");

            }else{

                $this->Flash->error("Could not remove investment!");

            }

        }

         //delete.php not needed

         return $this->redirect(['action' => 'index']);

    }

    public function edit($id){//id is recieved from index

        //get investments from model

      $investmentTable = $this->fetchTable('Investments');

      //get the first row by id ($id)

      $investmentToEdit = $investmentTable->findById($id)->first();

      //$this->set('allInvestments',$allInvestments);

      //check if form is submitted

      if ($this->request->is(['post', 'put'])) {

        //patchEntity(), instead of create() this method is used to update entity

       $investmentToEdit = $investmentTable->patchEntity($investmentToEdit, $this->request->getData());

              //patch(edit) user with the data from the form, $this->request->getData()

       if ($investmentTable->save($investmentToEdit)) {//save will save changes

           $this->Flash->success("User has been updated!");

           return $this->redirect(['action' => 'index']);

       }

       else {

           $this->Flash->error("Could not update investment!");

       }

     }

     //send investmentToEdit populate input fields for edit.php (view)

     $this->set("investmentToEdit", $investmentToEdit);

    }

    public function share($id){//id = investment id

        $investmentsTable = $this->fetchTable('Investments');

        $investmentToShare = $investmentsTable->get($id);

        $this->set('investmentToShare',$investmentToShare);//send to view

        //retrieve tickers and list items into select for options

        $tickersTable = $this->fetchTable('Tickers');

        $query = $tickersTable->find('list');

        $selectTicker = $query->toArray();

        $usersTable = $this->fetchTable('Users');

        $query = $usersTable->find('list');

        $selectUser = $query->toArray();

        $this->set('selectTicker',$selectTicker);//send to view

        $this->set('selectUser',$selectUser);//send to view to populate

        if ($this->request->is(['post', 'put'])) {

            //$user = $this->request->getData('user\_id');//getting selected user id

            $data = $this->request->getData();//getting the rest of the form

            //$investmentToShare->set('user\_id', $user);

            $investmentToShare = $investmentsTable->newEntity($data);

            //pr($investmentToShare);

            if($investmentsTable->save($investmentToShare)){//push data

                $this->Flash->success("Investment shared!");

                echo "Investment shared!";

            }else{

                $this->Flash->error("Error: InvestmentsController -- Investment not sahred");

                echo "Error: InvestmentsController -- Investment not added";

            }

        }

    }

    public function like($id){

        $investmentsTable = $this->fetchTable('Investments');

        $investmentToEdit = $investmentsTable->get($id);

        $investmentToEdit->like\_counter =  ($investmentToEdit->like\_counter)+1;

        $investmentsTable->save($investmentToEdit);

        //redirect to index inside Investments controller

        $redirect = $this->request->getQuery('redirect', [

            'controller' => 'Investments',

            'action' => 'index',

        ]);

        return $this->redirect($redirect);

    }

}

LIKES CONTROLLER

<?php

namespace App\Controller;

use Cake\ORM\Locator\LocatorAwareTrait;

use Cake\Log\Log;

use Cake\Log\Engine\FileLog;

class LikesController extends AppController{

    public function index(){

    }

    public function like($id){

        $investmentsTable = $this->fetchTable('Investments');

        $investmentToEdit = $investmentsTable->get($id);

        //add these 2 ids into LikesTable

        $likesTable = $this->getTableLocator()->get('Likes');

        $getLikedTradeUser = $this->loggedInUser->get('id');//user who liked the trade

        $newlikes = $likesTable->newEmptyEntity();

        $newlikes=$likesTable->find()->where(['user\_id'==$getLikedTradeUser])->where(['investment\_id'==$id])->first();

        $newlikes = $likesTable->newEmptyEntity();

        $newlikes->investment\_id = $id;//trade selected for like/dislike

        $newlikes->user\_id = $getLikedTradeUser;//user who pressed the like

        $newlikes->liked = 1;

        $newlikes->dislike = 0;

        $investmentToEdit->like\_counter =  $investmentToEdit->like\_counter+1;

        if ($likesTable->save($newlikes)) {

            echo "Like added!";

        }

        if ($investmentsTable->save($investmentToEdit)) {

            echo "Updated Investments!";

            Log::setConfig('likes', [

                'className' => FileLog::class,

                'path' => LOGS,

                'levels' => ['info'],

                'scopes' => ['likes'],

                'file' => 'likes.log',

            ]);

            Log::info('Logging: userid={user}, {message}, ip={ipaddress}, tradeid={trade\_id}', ['scope' => ['likes'], 'user' => $this->loggedInUser->get('id'),

            'ipaddress' => $this->request->clientIp(),

            'message'=> 'LikesController like() was successful',

            'trade\_id'=> $id]);

        }else{

            Log::setConfig('likes', [

                'className' => FileLog::class,

                'path' => LOGS,

                'levels' => ['error'],

                'scopes' => ['likes'],

                'file' => 'likes.log',

            ]);

            Log::error('Logging: userid={user}, {message}, ip={ipaddress}, tradeid={trade\_id}', ['scope' => ['likes'], 'user' => $this->loggedInUser->get('id'),

            'ipaddress' => $this->request->clientIp(),

            'message'=> 'LikesController like() was unsuccessful',

            'trade\_id'=> $id]);

        }

        $redirect = $this->request->getQuery('redirect', [

            'controller' => 'Investments',

            'action' => 'index',

        ]);

        return $this->redirect($redirect);

    }

    public function dislike($id){

        $investmentsTable = $this->fetchTable('Investments');

        $investmentToEdit = $investmentsTable->get($id);

        //add these 2 ids into LikesTable

        $likesTable = $this->getTableLocator()->get('Likes');

        $getLikedTradeUser = $this->loggedInUser->get('id');//user who liked the trade

        $newlikes = $likesTable->newEmptyEntity();

        $newlikes=$likesTable->find()->where(['user\_id'==$getLikedTradeUser])->where(['investment\_id'==$id])->first();

        $newlikes = $likesTable->newEmptyEntity();

        $newlikes->investment\_id = $id;//trade selected for like/dislike

        $newlikes->user\_id = $getLikedTradeUser;//user who pressed the like

        $newlikes->liked = 0;

        $newlikes->dislike = 1;

        $investmentToEdit->like\_counter =  $investmentToEdit->like\_counter-1;

        if ($likesTable->save($newlikes)) {

            echo "Like added!";

        }

        if ($investmentsTable->save($investmentToEdit)) {

            echo "Updated Investments!";

            Log::setConfig('likes', [

                'className' => FileLog::class,

                'path' => LOGS,

                'levels' => ['info'],

                'scopes' => ['likes'],

                'file' => 'likes.log',

            ]);

            Log::info('Logging: userid={user}, {message}, ip={ipaddress}, tradeid={trade\_id}', ['scope' => ['likes'], 'user' => $this->loggedInUser->get('id'),

            'ipaddress' => $this->request->clientIp(),

            'message'=> 'LikesController dislike() was successful',

            'trade\_id'=> $id]);

        }else{

            Log::setConfig('likes', [

                'className' => FileLog::class,

                'path' => LOGS,

                'levels' => ['error'],

                'scopes' => ['likes'],

                'file' => 'likes.log',

            ]);

            Log::error('Logging: userid={user}, {message}, ip={ipaddress}, tradeid={trade\_id}', ['scope' => ['likes'], 'user' => $this->loggedInUser->get('id'),

            'ipaddress' => $this->request->clientIp(),

            'message'=> 'LikesController dislike() was unsuccessful',

            'trade\_id'=> $id]);

        }

        $redirect = $this->request->getQuery('redirect', [

            'controller' => 'Investments',

            'action' => 'index',

        ]);

        return $this->redirect($redirect);

    }

}

USERS CONTROLLER

<?php

namespace App\Controller;

use Cake\ORM\Locator\LocatorAwareTrait;

use Cake\Log\Log;

use Cake\Log\Engine\FileLog;

class UsersController extends AppController{

public function beforeFilter(\Cake\Event\EventInterface $event)

{

   parent::beforeFilter($event);

   // Configure the login action to not require authentication, preventing

   // the infinite redirect loop issue

   $this->Authentication->addUnauthenticatedActions(['login','add','loginWithGoogle','google']);

      //allow unauthenicated users to access pages inside parameter -->login

}

public function login()

{

    $this->request->allowMethod(['get', 'post']);

    $result = $this->Authentication->getResult();

    // regardless of POST or GET, redirect if user is logged in

    if ($result->isValid()) {

         Log::setConfig('users', [

            'className' => FileLog::class,

            'path' => LOGS,

            'levels' => ['info'],

            'scopes' => ['users'],

            'file' => 'users.log',

        ]);

        Log::info('Logging: userid={user}, {message}, ip={ipaddress}', ['scope' => ['users'], 'user' => $this->loggedInUser->get('id'),

        'ipaddress' => $this->request->clientIp(),

        'message'=> 'UsersController login() was successful']);

        $redirect = $this->request->getQuery('redirect', [

            'controller' => 'Investments',

            'action' => 'index',

        ]);

        return $this->redirect($redirect);

    }

    // display error if user submitted and authentication failed

    if ($this->request->is('post') && !$result->isValid()) {

        $this->Flash->error('Invalid username or password');

        Log::setConfig('users', [

         'className' => FileLog::class,

         'path' => LOGS,

         'levels' => ['error'],

         'scopes' => ['users'],

         'file' => 'users.log',

     ]);

     Log::error('Logging: userid={user}, {message}, ip={ipaddress}', ['scope' => ['users'], 'user' => $this->loggedInUser->get('id'),

     'ipaddress' => $this->request->clientIp(),

     'message'=> 'UsersController login() was unsuccessful']);

    }

}

    public function logout()

   {

    $result = $this->Authentication->getResult();

    // regardless of POST or GET, redirect if user is logged in

    if ($result->isValid()) {

        $this->Authentication->logout();

        Log::setConfig('users', [

         'className' => FileLog::class,

         'path' => LOGS,

         'levels' => ['info'],

         'scopes' => ['users'],

         'file' => 'users.log',

     ]);

     Log::info('Logging: userid={user}, {message}, ip={ipaddress}', ['scope' => ['users'], 'user' => $this->loggedInUser->get('id'),

     'ipaddress' => $this->request->clientIp(),

     'message'=> 'UsersController logout() was successful']);

      return $this->redirect(['controller' => 'Users', 'action' => 'login']);

    }else{

      Log::setConfig('users', [

         'className' => FileLog::class,

         'path' => LOGS,

         'levels' => ['error'],

         'scopes' => ['users'],

         'file' => 'users.log',

     ]);

     Log::error('Logging: userid={user}, {message}, ip={ipaddress}', ['scope' => ['users'], 'user' => $this->loggedInUser->get('id'),

     'ipaddress' => $this->request->clientIp(),

     'message'=> 'UsersController logout() was unsuccessful']);

    }

   }

   public function add()

   {

      if ($this->request->is("post")) {

         $usersTable = $this->fetchTable('Users');

         $data = $this->request->getData();//get data from form, store it in $data

          //strip\_tags (for additional security)

          $data['first\_name'] = strip\_tags($this->request->getData('first\_name'));

          $data['last\_name'] = strip\_tags($this->request->getData('last\_name'));

          $newUser = $usersTable->newEntity($data);

         if ($usersTable->save($newUser)) {//save() will save the changes

            $this->Flash->success("User has been saved!");

            return $this->redirect(['action' => 'login']);

         }

         else {

             $errors\_messages="";

             $errors = $newUser->getErrors();//get list of errors from UsersTable table validator

             foreach ($errors as $value){//go through each error in the list

                $errors\_messages .= array\_values($value)[0]."<br>;";//.= , this appends to string,

                                                                   //errors\_messages

             }

             echo "{$errors\_messages}";//OR

             //However flash erros is not working :(

             //$this->Flash->error("Something wrong happened during user insertion<br>$error\_messages",

                //['escape'=>false]);//'escape'=>false] , to enable functionality for <br> tags

          }

      }

   }

   public function google(){

   }

   public function userlist(){

      $usersTable = $this->fetchTable('Users');

      $allUsers = $usersTable->find('all')->toArray();

      $this->set('allUsers', $allUsers);

   }

}

***ENTITY***

USER

<?php

//reference: https://book.cakephp.org/4/en/tutorials-and-examples/cms/authentication.html

namespace App\Model\Entity;

use Authentication\PasswordHasher\DefaultPasswordHasher; // Add this line

use Cake\ORM\Entity;

class User extends Entity

{

    // Code from bake.

    // Add this method

    protected function \_setPassword(string $password) : ?string

    {

        if (strlen($password) > 0) {

            return (new DefaultPasswordHasher())->hash($password);

        }

    }

}

***TABLE***

INVESTMENTS

<?php

namespace App\Model\Table;

use Cake\ORM\Table;

use Cake\Validation\Validator;

class InvestmentsTable extends Table{

    //relationships: https://book.cakephp.org/4/en/orm/associations.html

    public function initialize(array $config):void

    {

        $this->belongsTo('Tickers');

        $this->belongsTo('Users');

        $this->hasOne('Likes');

        $this->addBehavior('Timestamp');

    }

    public function validationDefault(Validator $validator): Validator

    {

         $validator

            ->notEmptyString('share',"share name is required")

            ->add('length', 'custom', [

                'rule' => function ($value, $context) {

                    if (!$value) {

                        return false;

                    }

                    if ($value < 1) {

                        return 'Error: min share  is 1';

                    }

                    return true;

                }])

            ->notEmptyString('bought\_at',"bought at is required");

         return $validator;

    }

}

LIKES

<?php

namespace App\Model\Table;

use Cake\ORM\Table;

use Cake\Validation\Validator;

class LikesTable extends Table

{

    //Creating A Default Validation Set

    //reference: https://book.cakephp.org/4/en/orm/validation.html

    /\*

    [Model validates when adding and editing user]

    Request data will be validated.

    Valid data will be set into the entity, while fields that failed validation will be excluded.

    \*/

    //reference: https://book.cakephp.org/4/en/orm/associations.html#

    public function initialize(array $config):void

    {

    }

}

TICKERS

<?php

namespace App\Model\Table;

use Cake\ORM\Table;

class TickersTable extends Table{

    public function initialize(array $config):void

    {

        $this->hasMany('Investments');

        //

        $this->setDisplayField('ticker');

    }

}

USERS

<?php

namespace App\Model\Table;

use Cake\ORM\Table;

use Cake\Validation\Validator;

class UsersTable extends Table

{

    //Creating A Default Validation Set

    //reference: https://book.cakephp.org/4/en/orm/validation.html

    /\*

    [Model validates when adding and editing user]

    Request data will be validated.

    Valid data will be set into the entity, while fields that failed validation will be excluded.

    \*/

    //reference: https://book.cakephp.org/4/en/orm/associations.html#

    public function initialize(array $config):void

    {

        $this->hasMany('Investments');

        $this->hasOne('Likes');

        $this->setDisplayField('email');

    }

}

***VIEW HELPER***

APPVIEW

<?php

declare(strict\_types=1);

/\*\*

 \* CakePHP(tm) : Rapid Development Framework (https://cakephp.org)

 \* Copyright (c) Cake Software Foundation, Inc. (https://cakefoundation.org)

 \*

 \* Licensed under The MIT License

 \* Redistributions of files must retain the above copyright notice.

 \*

 \* @copyright Copyright (c) Cake Software Foundation, Inc. (https://cakefoundation.org)

 \* @link      https://cakephp.org CakePHP(tm) Project

 \* @since     3.0.0

 \* @license   https://opensource.org/licenses/mit-license.php MIT License

 \*/

namespace App\View;

use Cake\View\View;

/\*\*

 \* Application View

 \*

 \* Your application's default view class

 \*

 \* @link https://book.cakephp.org/4/en/views.html#the-app-view

 \*/

class AppView extends View

{

    /\*\*

     \* Initialization hook method.

     \*

     \* Use this method to add common initialization code like loading helpers.

     \*

     \* e.g. `$this->loadHelper('Html');`

     \*

     \* @return void

     \*/

    public function initialize(): void

    {

        $this->loadHelper('Authentication.Identity');

    }

}

APPLICATION.PHP

<?php

declare(strict\_types=1);

/\*\*

 \* CakePHP(tm) : Rapid Development Framework (https://cakephp.org)

 \* Copyright (c) Cake Software Foundation, Inc. (https://cakefoundation.org)

 \*

 \* Licensed under The MIT License

 \* For full copyright and license information, please see the LICENSE.txt

 \* Redistributions of files must retain the above copyright notice.

 \*

 \* @copyright Copyright (c) Cake Software Foundation, Inc. (https://cakefoundation.org)

 \* @link      https://cakephp.org CakePHP(tm) Project

 \* @since     3.3.0

 \* @license   https://opensource.org/licenses/mit-license.php MIT License

 \*/

namespace App;

use Cake\Core\Configure;

use Cake\Core\ContainerInterface;

use Cake\Datasource\FactoryLocator;

use Cake\Error\Middleware\ErrorHandlerMiddleware;

use Cake\Http\BaseApplication;

use Cake\Http\Middleware\BodyParserMiddleware;

use Cake\Http\Middleware\CsrfProtectionMiddleware;

use Cake\Http\MiddlewareQueue;

use Cake\ORM\Locator\TableLocator;

use Cake\Routing\Middleware\AssetMiddleware;

use Cake\Routing\Middleware\RoutingMiddleware;

// In src/Application.php add the following imports

use Authentication\AuthenticationService;

use Authentication\AuthenticationServiceInterface;

use Authentication\AuthenticationServiceProviderInterface;

use Authentication\Middleware\AuthenticationMiddleware;

use Cake\Routing\Router;

use Psr\Http\Message\ServerRequestInterface;

/\*\*

 \* Application setup class.

 \*

 \* This defines the bootstrapping logic and middleware layers you

 \* want to use in your application.

 \*

 \* // in src/Application.php

 \*/

class Application extends BaseApplication implements AuthenticationServiceProviderInterface

{

    /\*\*

     \* Load all the application configuration and bootstrap logic.

     \*

     \* @return void

     \*/

    public function bootstrap(): void

    {

        // Call parent to load bootstrap from files.

        parent::bootstrap();

        if (PHP\_SAPI === 'cli') {

            $this->bootstrapCli();

        } else {

            FactoryLocator::add(

                'Table',

                (new TableLocator())->allowFallbackClass(false)

            );

        }

        /\*

         \* Only try to load DebugKit in development mode

         \* Debug Kit should not be installed on a production system

         \*/

        if (Configure::read('debug')) {

            $this->addPlugin('DebugKit');

        }

        // Load more plugins here

    }

    /\*\*

     \* Setup the middleware queue your application will use.

     \*

     \* @param \Cake\Http\MiddlewareQueue $middlewareQueue The middleware queue to setup.

     \* @return \Cake\Http\MiddlewareQueue The updated middleware queue.

     \*/

    public function middleware(MiddlewareQueue $middlewareQueue): MiddlewareQueue

    {

        $csrf = new CsrfProtectionMiddleware(['httponly'=>true]);

    $csrf->skipCheckCallback(function($request) {            // Token check will be skipped when callback returns `true`.

        $controller = $request->getParam('controller');

        $action = $request->getParam('action');

        if (is\_null($controller) || is\_null($action)) {

            return false;

        }

        if (strcmp($controller,'Apis') == 0) {    // Skip token check for API URLs.

            return true;

        }

        return false;

    });

        $middlewareQueue

            ->add(new ErrorHandlerMiddleware(Configure::read('Error')))

            ->add(new RoutingMiddleware($this))

            ->add(new AssetMiddleware([

                'cacheTime' => Configure::read('Asset.cacheTime'),

            ]))

            ->add(new RoutingMiddleware($this))

            ->add(new AuthenticationMiddleware($this))

            ->add(new BodyParserMiddleware())

            ->add(new CsrfProtectionMiddleware([

                'httponly' => true,

            ]));

        return $middlewareQueue;

    }

    public function getAuthenticationService(ServerRequestInterface $request): AuthenticationServiceInterface

{

    $authenticationService = new AuthenticationService([

        'unauthenticatedRedirect' => Router::url('/users/login'),

        'queryParam' => 'redirect',

    ]);

    // Load identifiers, ensure we check email and password fields

    $authenticationService->loadIdentifier('Authentication.Password', [

        'fields' => [

            'username' => 'email',

            'password' => 'password',

        ]

    ]);

    // Load the authenticators, you want session first

    $authenticationService->loadAuthenticator('Authentication.Session');

    // Configure form data check to pick email and password

    $authenticationService->loadAuthenticator('Authentication.Form', [

        'fields' => [

            'username' => 'email',

            'password' => 'password',

        ],

        'loginUrl' => Router::url('/users/login'),

    ]);

    return $authenticationService;

}

    /\*\*

     \* Register application container services.

     \*

     \* @param \Cake\Core\ContainerInterface $container The Container to update.

     \* @return void

     \* @link https://book.cakephp.org/4/en/development/dependency-injection.html#dependency-injection

     \*/

    public function services(ContainerInterface $container): void

    {

    }

    /\*\*

     \* Bootstrapping for CLI application.

     \*

     \* That is when running commands.

     \*

     \* @return void

     \*/

    protected function bootstrapCli(): void

    {

        $this->addOptionalPlugin('Cake/Repl');

        $this->addOptionalPlugin('Bake');

        $this->addPlugin('Migrations');

        // Load more plugins here

    }

}

***TEMPLATES – INVESTMENTS***

ADD

<h1>Add.php - Investments Controller</h1>

<?php

echo $this->Form->create();

//select option

echo $this->Form->select('ticker\_id', ['options' => $selectTicker]);

echo $this->Form->control('share', ['required' => true,

                                    'class' => 'form-control mb-3']);

echo $this->Form->control('bought\_at', ['required' => true,

                                        'class' => 'form-control mb-3']);

echo $this->Form->control('note', ['required' => false,

                                'class' => 'form-control mb-3']);

echo $this->Form->control('privacy', ['type' => 'checkbox']);

                        //i.e. 0 for false, 1-255 (preferably 1) for true.PUBLIC==1

echo $this->Form->submit('Subimt');

EDIT

<h2>Investments - Edit</h2>

<p>Fill in this form to edit a investment</p>

<div class="row">

    <div class="col-6 offset-3">

        <?php

            //https://book.cakephp.org/4/en/views/helpers/form.html

            //Form Helper

            echo $this->Form->create($investmentToEdit);//being recieved from the controller

                                    //this will populate the form with investment of which id == to id

            echo $this->Form->control("share", [

                                                        'placeholder' => 'Enter your share',

                                                        'label' => 'Sale',

                                                        'class' => 'form-control mb-3'

                                                    ]);

            echo $this->Form->control("bought\_at", [

                                                        'placeholder' => 'Enter your price',

                                                        'label' => 'Bought At',

                                                        'class' => 'form-control mb-3'

                                                    ]);

            echo $this->Form->control("note", [

                                                        'placeholder' => 'Enter your note',

                                                        'label' => 'Note',

                                                        'class' => 'form-control mb-3'

                                                    ]);

            echo $this->Form->control('privacy', ['type' => 'checkbox']);

            echo $this->Form->submit("Edit Investment", ['class' => 'btn btn-primary mt-3']);

            echo $this->Form->end();

        ?>

    </div>

</div>

INDEX

<?php

$totalPortfolio = 0;

$getUsername = $user\_username;//get logged in email

$getSession =  $user\_session;//get logged in id

echo "<div class='row'>";

echo "<h2>Your trades ($getUsername)</h2>";

if (count($allInvestments) >= 1){

        foreach($allInvestments as $investment){

            if($investment->user\_id == $getSession ){

                echo "<div class='col-sm-6'>

                <div class='card'>

                <div class='card-body'><h3>";

                echo $investment->ticker->ticker;//show name

                echo "</h3><br>";

                echo "Shares $investment->share";

                echo " at $investment->bought\_at";

                echo "<br>";

                echo "Total Spent: ". $investment->bought\_at \* $investment->share;

                echo "<br>";

                echo "Notes: ";

                if($investment->note == ""){

                    echo "N/A";

                }else{

                    echo "$investment->note";

                }

                echo "<br>";

                $getTime =  $investment->created;

                echo " at ".$getTime->i18nFormat('dd-MMM-yyyy HH:mm');

                echo "<br>";

                $deleteLink = $this->Url->build("/investments/delete/".$investment->id);

                echo '<td><a href="'.$deleteLink.'" class="btn btn-danger">Delete</a>';

                $editLink = $this->Url->build("/investments/edit/".$investment->id);

                echo '<td><a href="'.$editLink.'" class="btn btn-warning">Edit</a>';

                $shareLink = $this->Url->build("/investments/share/".$investment->id);

                echo '<td><a href="'.$shareLink.'" class="btn btn-warning">Share</a>';

                $totalPortfolio +=$investment->bought\_at \* $investment->share;

                echo "</div></div></div><br>";

            }

        }

        echo "</div>";

    }

    echo "<hr><h2>Public trades</h2>";

    echo "<div class='row'>";

if (count($allInvestments) >= 1){//NOTE IF USER IS ADMIN ADD FULL NAME TO THE CARD

        foreach($allInvestments as $investment){

            if($investment->privacy >0 ){

                echo "<div class='col-sm-6'>

                <div class='card'>

                <div class='card-body'><h3>";

                $getUser = $usersTable->findById($investment->user\_id)->first();

                $fullname="";

                if($getUsername == 'admin@gmail.com'){

                    $fullname=$getUser->first\_name." ".$getUser->last\_name.", bought ";

                }

                echo $fullname.$investment->ticker->ticker;//show name

                echo "</h3><br>";

                echo "Bought shares $investment->share";

                echo " at $investment->bought\_at";

                echo "<br>";

                echo "Total Spent: ". $investment->bought\_at \* $investment->share;

                echo "<br>";

                echo "Notes: ";

                if($investment->note == ""){

                    echo "N/A";

                }else{

                    echo "$investment->note";

                }

                echo "<br>";

                $getTime =  $investment->created;

                echo " at ".$getTime->i18nFormat('dd-MMM-yyyy HH:mm');

                echo "<br>Likes: $investment->like\_counter";

                echo "<br>";

                //$isLiked = $likesTable->find()->where(['investment\_id' => $investment->id],['user\_id' => $investment->user\_id])->first();

                $dislikeLink = $this->Url->build("/likes/dislike/".$investment->id);

                echo '<td><a href="'.$dislikeLink.'" class="btn btn-danger">Add Dislike</a>';

                $likeLink = $this->Url->build("/likes/like/".$investment->id);

                echo '<td><a href="'.$likeLink.'" class="btn btn-danger">Add Like</a>';

                if($getUsername == 'admin@gmail.com' ){

                    $deleteLink = $this->Url->build("/investments/delete/".$investment->id);

                    echo '<td><a href="'.$deleteLink.'" class="btn btn-danger">Delete</a>';

                }

                echo "</div></div></div><br>";

            }

        }

    }

if($getUsername == 'admin@gmail.com' ){

    echo "<hr><h2>Private trades</h2>";

    echo "<div class='row'>";

    if (count($allInvestments) >= 1){//NOTE IF USER IS ADMIN ADD FULL NAME TO THE CARD

        foreach($allInvestments as $investment){

            if($investment->privacy ==0 ){

                echo "<div class='col-sm-6'>

                <div class='card'>

                <div class='card-body'><h3>";

                $getUser = $usersTable->findById($investment->user\_id)->first();

                $fullname="";

                if($getUsername == 'admin@gmail.com'){

                    $fullname=$getUser->first\_name." ".$getUser->last\_name.", bought ";

                }

                echo $fullname.$investment->ticker->ticker;//show name

                echo "</h3><br>";

                echo "Bought shares $investment->share";

                echo " at $investment->bought\_at";

                echo "<br>";

                echo "Total Spent: ". $investment->bought\_at \* $investment->share;

                echo "<br>";

                echo "Notes: ";

                if($investment->note == ""){

                    echo "N/A";

                }else{

                    echo "$investment->note";

                }

                echo "<br>";

                $getTime =  $investment->created;

                echo " at ".$getTime->i18nFormat('dd-MMM-yyyy HH:mm');

                echo "<br>Likes: $investment->like\_counter";

                echo "<br>";

                if($getUsername == 'admin@gmail.com' ){

                    $deleteLink = $this->Url->build("/investments/delete/".$investment->id);

                    echo '<td><a href="'.$deleteLink.'" class="btn btn-danger">Delete</a>';

                }

                echo "</div></div></div><br>";

            }

        }

    }

}

    echo "<div class='alert alert-primary' role='alert'>

        Your total Porfolio: $ $totalPortfolio</div>";

SHARE

<h1>Share.php - Investments Controller</h1>

<div class="row">

    <div class="col-6 offset-3">

    <?php

    echo $this->Form->create($investmentToShare);//being recieved from the controller

        //this will populate the form with investment of which is == to id

    //select option

    echo $this->Form->select('user\_id', ['options' => $selectUser ]);

    echo $this->Form->select('ticker\_id', ['options' => $selectTicker]);

    echo $this->Form->control('share', ['disabled' => false,

                                        'class' => 'form-control mb-3']);

    echo $this->Form->control('bought\_at', ['disabled' => false,

                                        'class' => 'form-control mb-3']);

    echo $this->Form->control('note', ['disabled' => false,

                                        'class' => 'form-control mb-3']);

    echo $this->Form->control('privacy', ['type' => 'checkbox','disabled' => false,]);

                            //i.e. 0 for false, 1-255 (preferably 1) for true.

    echo $this->Form->submit('Share', ['class' => 'btn btn-primary mt-3']);

    echo $this->Form->end();

    ?>

    </div>

</div>

***LAYOUT***

DEFAULT

<!DOCTYPE html>

<html>

<head>

    <?= $this->Html->charset() ?>

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>

        <?= $this->fetch('title') ?>

    </title>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">

    <?= $this->fetch('meta') ?>

    <?= $this->fetch('css') ?>

    <?= $this->fetch('script') ?>

</head>

<body>

    <div class="container">

        <ul class="nav">

            <li class="nav-item">

                <a class="nav-link active" aria-current="page" href="<?=$this->Url->build(['controller'=>'Investments','action'=>'index']);?>">View Investment</a>

                                                                     <!-- Url->build("/"), go to base class-->

            </li>

            <li class="nav-item">

                <a class="nav-link active" aria-current="page" href="<?=$this->Url->build(['controller'=>'Investments','action'=>'add']);?>">Add Investment</a>

            </li>

            <li class="nav-item">

                <a class="nav-link active" aria-current="page" href="<?=$this->Url->build(['controller'=>'Users','action'=>'userlist']);?>">View Users</a>

            </li>

            <li class="nav-item">

                <a class="nav-link active" aria-current="page" href="<?=$this->Url->build(['controller'=>'Users','action'=>'logout']);?>">Logout</a>

            </li>

        </ul>

        <hr>

        <?= $this->Flash->render() ?>

        <?= $this->fetch('content') ?>

        <footer class="text-center text-muted">

            <hr>

            <p>My CakePHP App - Jake Bugeja</p>

        </footer>

    </div>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p" crossorigin="anonymous"></script>

</body>

</html>

LIKES TEMPLATE

DISLIKE

<p>dislike.php</p>

LIKE

<p>Like.php - view</p>

USERS TEMPLATE

ADD

<h2>Add a new User</h2>

<?php

//reference: https://book.cakephp.org/4/en/viewas/helpers/form.html

//reference: lesson 13 2020, 6.2A

    //creating the form tag , <form>

    echo $this->Form->create();

    echo "<br><br>";

    //Creating Form Controls

    echo $this->Form->control('first\_name', ['required' => false,

                                            'class' => 'form-control mb-3']);

    echo $this->Form->control('last\_name', ['required' => false,

                                            'class' => 'form-control mb-3']);

    echo "<br>";

    echo $this->Form->control('email', ['required' => true,

                                        'class' => 'form-control mb-3']);

    echo $this->Form->control("password",['required' => true,

                                        'class' => 'form-control mb-3']);

    //Creating Buttons and Submit Elements

    echo $this->Form->submit("Add user",['class' => "btn btn-primary mt-3 btn-block"]);

    echo $this->Form->end();

LOGIN

<script src="google.js"></script>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<!--IF imports do not work <script src="https://www.gstatic.com/firebasejs/8.0.0/firebase-auth.js"></script>-->

<script type="module">

  // Import the functions you need from the SDKs you need

  import { initializeApp } from "https://www.gstatic.com/firebasejs/9.6.2/firebase-app.js";

  import { } from 'firebase/auth';

  // TODO: Add SDKs for Firebase products that you want to use

  // https://firebase.google.com/docs/web/setup#available-libraries

  // Your web app's Firebase configuration

  const firebaseConfig = {

    apiKey: "AIzaSyDI9YPrbhnlz\_vjrW0fdbl88Xag5sIw8uQ",

    authDomain: "greek-investment-app.firebaseapp.com",

    projectId: "greek-investment-app",

    storageBucket: "greek-investment-app.appspot.com",

    messagingSenderId: "881961358640",

    appId: "1:881961358640:web:fbbb7795798271c1c8a4f5"

  };

  // Initialize Firebase

  const app = initializeApp(firebaseConfig);

  document.getElementById('googlebttn').addEventListener('click',GoogleLogin)

  //document.getElementById (FOR THE LOGOUT)

  function GoogleLogin(){

      alert("TTT");

  }

</script>

<script>

//jquery

</script>

<div class="users form">

    <h3>Login</h3>

    <button id="googlebttn">Login with Google</button>

    <?= $this->Form->create() ?>

    <fieldset>

        <legend><?= \_\_('Please enter your username and password') ?></legend>

        <?= $this->Form->control('email', ['required' => true]) ?>

        <?= $this->Form->control('password', ['required' => true]) ?>

    </fieldset>

    <?= $this->Form->submit(\_\_('Login')); ?>

    <?= $this->Form->end() ?>

    <?= $this->Html->link("Add User", ['action' => 'add']) ?>

</div>

USERLIST

<?php

echo "<div class='row'>";

if (count($allUsers) >= 1){

        foreach($allUsers as $user){

            echo "<div class='col-sm-6'>

            <div class='card'>

            <div class='card-body'><h3>";

            echo $user->first\_name.' '.$user->last\_name;//show name

            echo "</h3><br>";

            echo $user->email;

            echo "<br>User Signed in: ";

            $session =  $this->Identity->get('email');

            echo "$session";

            //ADMIN ONLY

            if($session == "admin@gmail.com"){

                $viewTradesLink = $this->Url->build("/investments/index/".$user->id);

                echo '<br><a href="'.$viewTradesLink.'" class="btn btn-danger">View Trades</a>';

            }

            echo "</div></div></div><br>";

        }

        echo "</div>";

    }