# 实现文档

## 前期规划及任务分工

| 任务标题 | 优先级 | 截止时间 | 负责人 | 参与成员 |
| --- | --- | --- | --- | --- |
| 数据库设计 | 普通 | | Su | Su |
| 网站数据整理 | 普通 | | Su | Su 万帆 & |
| 网站前端框架搭建 | 普通 | | 万帆 | 万帆 |
| 网站后端框架搭建 | 普通 | | & | & |
| 使用Gii生成相关代码 | 普通 | | Su | Su 万帆 & |
| 网站前端完善 | 普通 | | 万帆 | Su 万帆 & |
| 网站后端完善 | 普通 | | & | Su 万帆 & |

本次项目使用 Tower 进行任务的划分

## 小组人员及分工如下：

### 苏长昊：

整体框架构建
数据库设计
网站数据收集整理
参与前端代码的完善
参与后端代码的完善

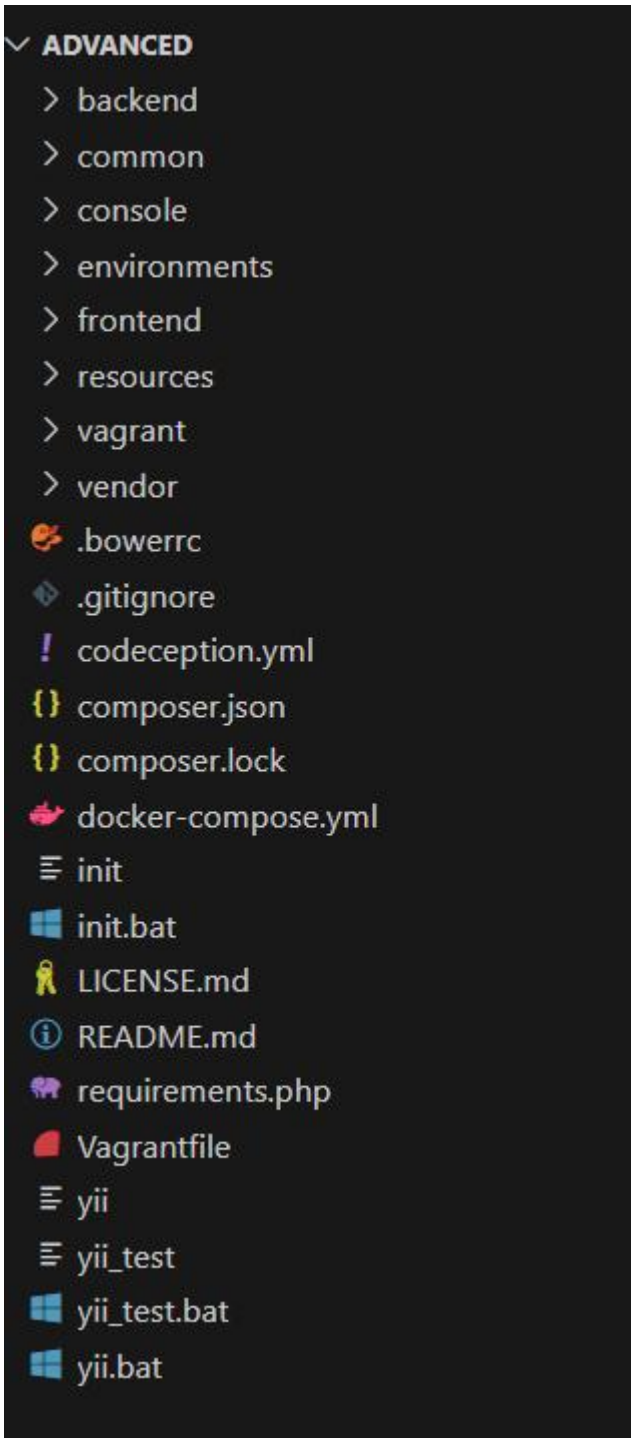
### 陈鹏：

整体框架构建
后端框架完善、细节优化
后端代码实现
参与数据收集与整理

### 张铮：

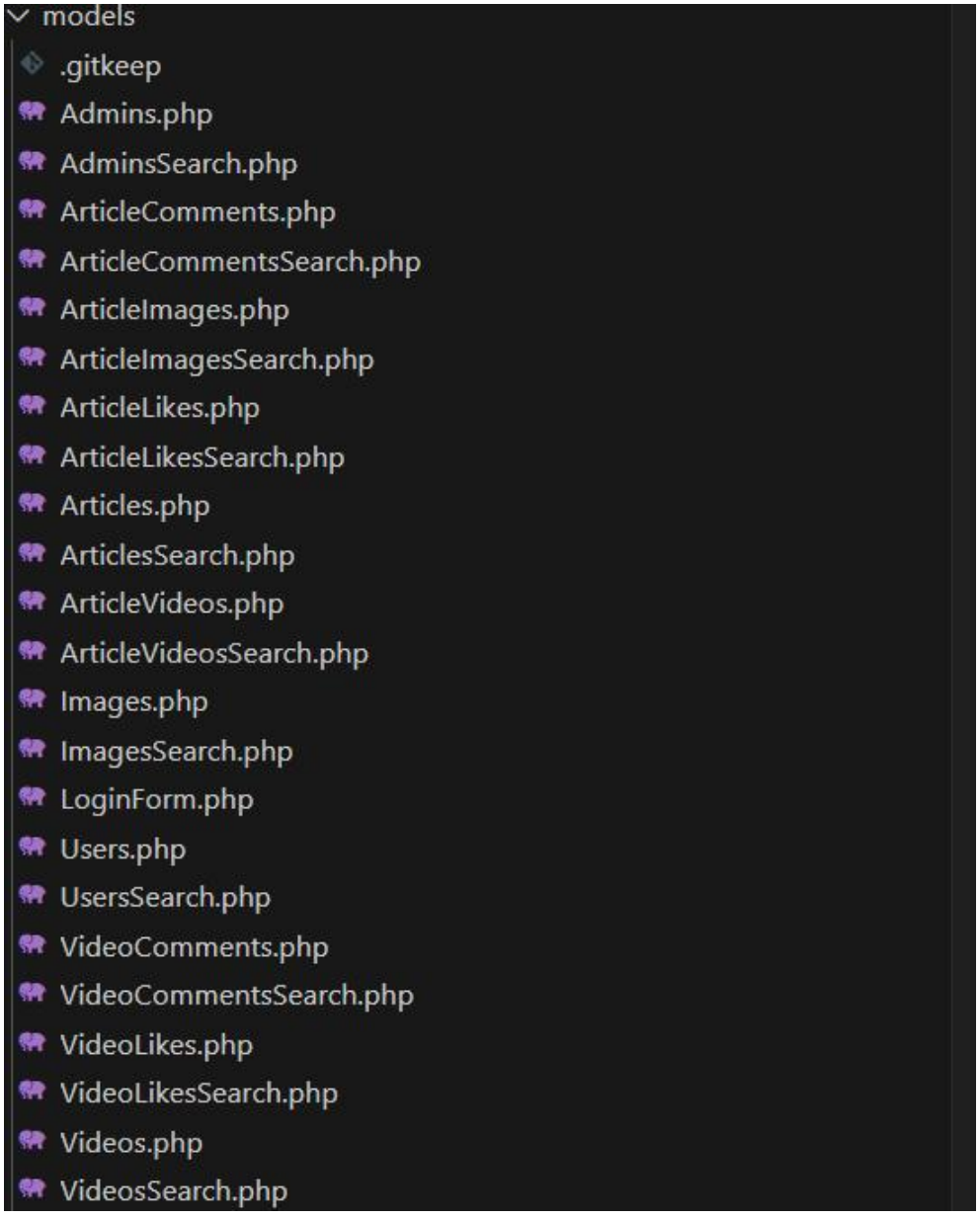
整体框架构建
前端框架完善、细节优化
前端代码实现
参与数据库设计

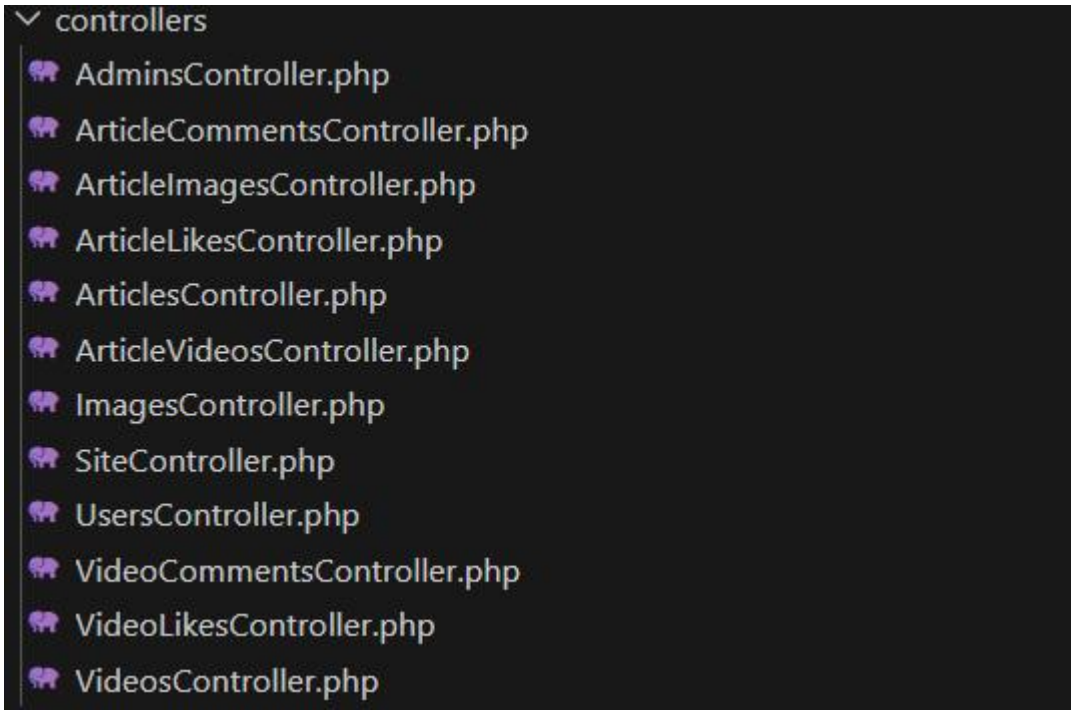
项目目录：

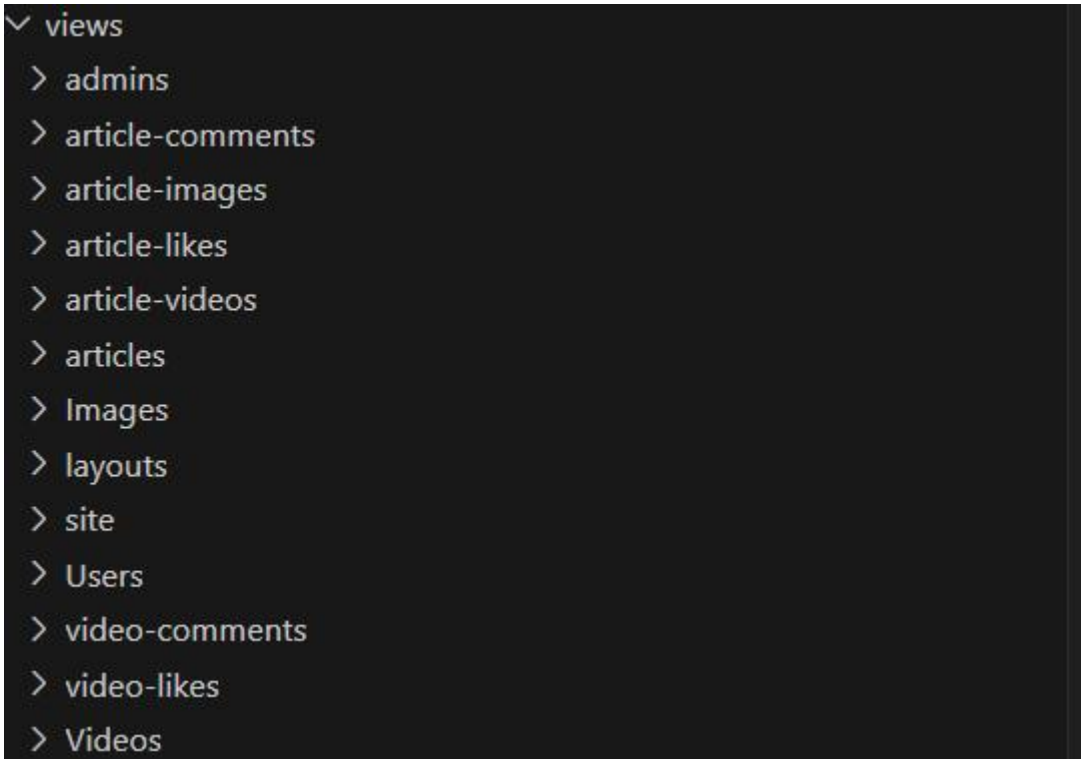



上图展示了本项目整体框架，backend 文件夹为后端代码，frontend 文件夹为前端代码，resources 保存了网站中的文章、图片、视频资源，common 文件夹中为项目相关的一些配置（如 config/main-local.php 是数据库配置文件）


backend
assets
config
controllers
models
runtime
tests
views
web

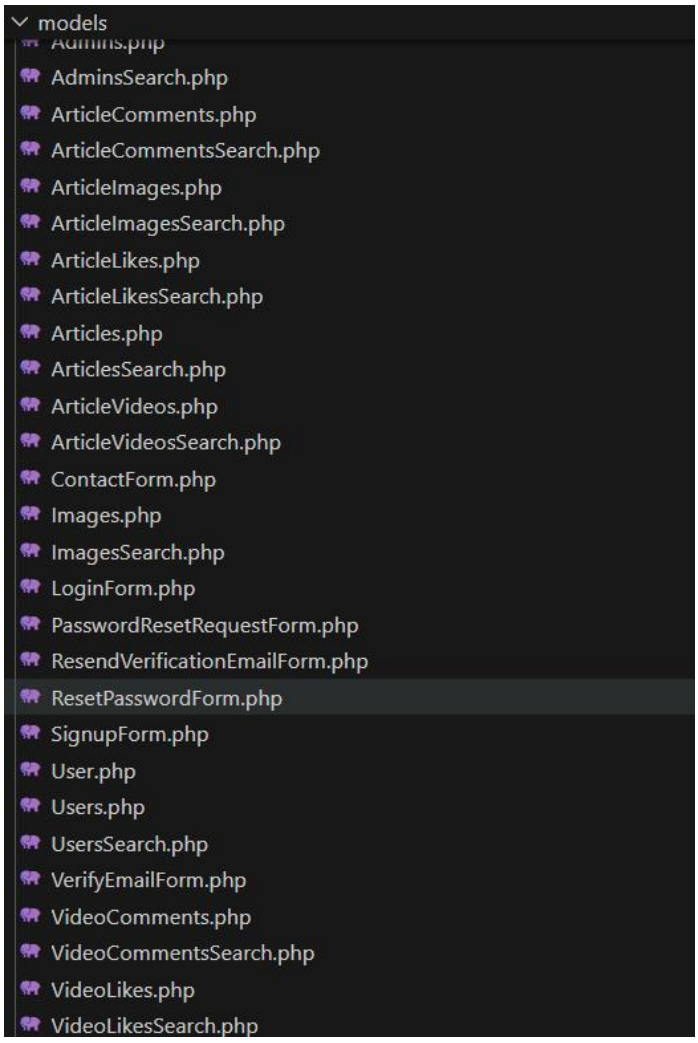

models
.gitkeep
Admins.php
AdminsSearch.php
ArticleComments.php
ArticleCommentsSearch.php
ArticleImages.php
ArticleImagesSearch.php
ArticleLikes.php
ArticleLikesSearch.php
Articles.php
ArticlesSearch.php
ArticleVideos.php
ArticleVideosSearch.php
Images.php
ImagesSearch.php
LoginForm.php
Users.php
UsersSearch.php
VideoComments.php
VideoCommentsSearch.php
VideoLikes.php
VideoLikesSearch.php
Videos.php
VideosSearch.php

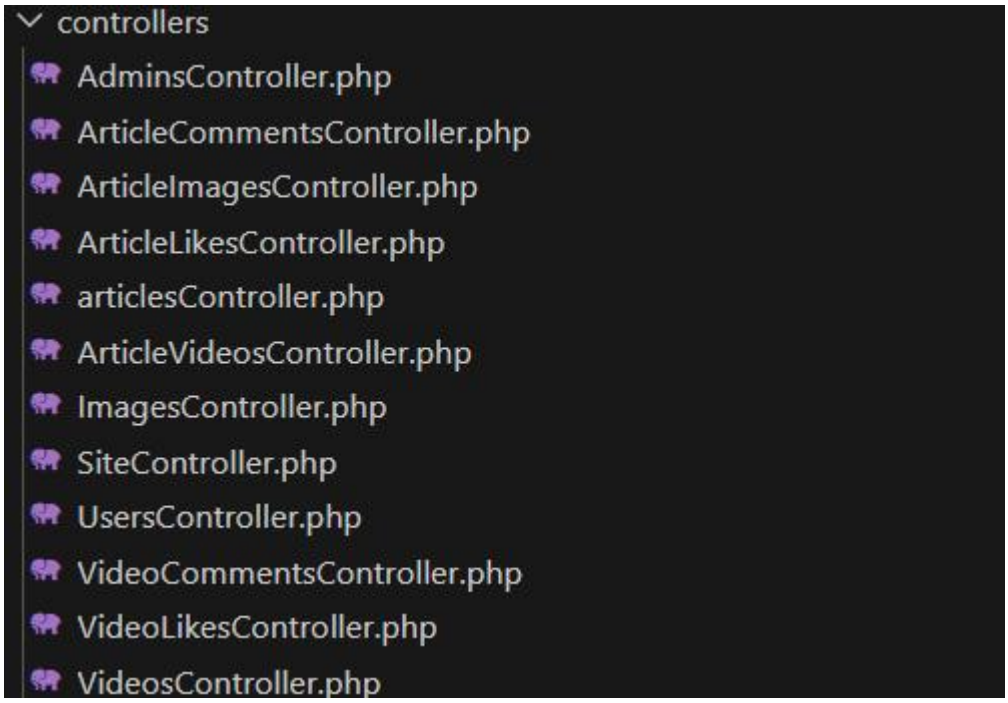


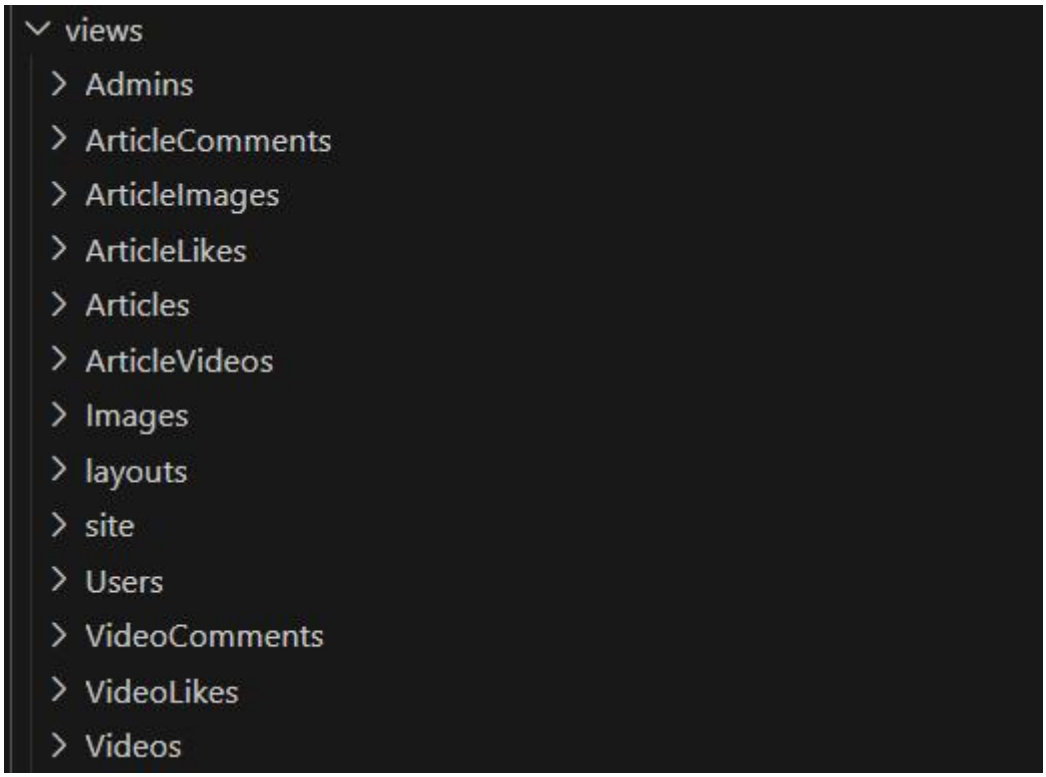




Backend 文件目录下 models 存放模型类文件，实现应用程序与数据库表间的映射；controllers 存放控制器类文件，负责处理用户请求，views 存放了网站的视图文件，完成网站后台管理的可视化功能


frontend 文件目录下 models 存放模型类文件，实现应用程序与数据库表间的映射；controllers 存放控制器类文件，负责处理用户请求，views 存放了网站的视图文件，完成网站前端的可视化功能






代码展示：

以文章部分前端代码为例

models/Articles

tableName 定义了模型所映射的数据库表名

```php
/**
 * {@inheritdoc}
 */
public static function tableName()
{
    return '{{%articles}}';
}
```

rules 定义了属性与规则，attributeLabels 定义了每个属性的标签，通常用于表单渲染和显示字段名称

```php
public function rules()
{
    return [
        [['title', 'content_url'], 'required'],
        [['created_at', 'updated_at'], 'safe'],
        [['title', 'content_url'], 'string', 'max' => 255]
    ];
}

/**
 * {@inheritdoc}
 */
public function attributeLabels()
{
    return [
        'id' => 'ID',
        'title' => 'Title',
        'content_url' => 'Content Url',
        'created_at' => 'Created At',
        'updated_at' => 'Updated At',
    ];
}
```

getArticleImages、getArticleVideos，定义了文章与图片、视频的关联

```php
/**
 * 获取文章关联的图片
 * @return \yii\db\ActiveQuery
 */
public function getArticleImages()
{
    return $this->hasMany(ArticleImages::class, ['article_id' => 'id'])->with('image');
}

/**
 * 获取文章关联的视频
 * @return \yii\db\ActiveQuery
 */
public function getArticleVideos()
{
    return $this->hasMany(ArticleVideos::class, ['article_id' => 'id'])->with('video');
}
```

actionIndex 方法：处理文章列表的展示，调用 ArticlesSearch 模型进行搜索并获取数据提供器 ($dataProvider)，最终渲染到视图 index 中

```php
public function actionIndex()
{
    $searchModel = new ArticlesSearch();
    $dataProvider = $searchModel->search(Yii::$app->request->queryParams);

    return $this->render('index', [
        'searchModel' => $searchModel,
        'dataProvider' => $dataProvider,
    ]);
}
```

actionView 方法显示单篇文章的详细信息，包括：获取文章模型及其相关数据（如图片和视频）。获取文章的所有评论，并按创建时间倒序排列。计算该文章的点赞数。处理新的评论提交：若用户提交新评论，ArticleComments 模型会保存评论数据（包含文章 ID、用户 ID、评论时间等）。最后，渲染 view 视图，并将文章、评论、点赞数量等数据传递给视图。

```php
public function actionView($id)
{
    $model = $this->findModel($id);
    $images = $model->images; // 获取关联的图片数据
    $videos = $model->videos; // 获取关联的视频数据
    // Fetch comments
    $comments = ArticleComments::find()->where(['article_id' => $id])->orderBy(['created_at' => SORT_DESC])->all();

    // Count likes
    $likeCount = ArticleLikes::find()->where(['article_id' => $id])->count();

    // Handle new comment submission
    $newComment = new ArticleComments();
    if ($newComment->load(Yii::$app->request->post())) {
        $newComment->article_id = $id;
        $newComment->user_id = Yii::$app->user->id ?? null; // Assign user ID if logged in
        $newComment->created_at = date('Y-m-d H:i:s');
        if ($newComment->save()) {
            return $this->redirect(['view', 'id' => $id]);
        }
    }

    return $this->render('view', [
        'model' => $model,
        'images' => $images,
        'videos' => $videos,
        'comments' => $comments,
        'newComment' => $newComment,
        'likeCount' => $likeCount,
    ]);
}
```

点赞文章（actionLike）：actionLike($id) 方法处理用户对文章的点赞：检查用户是否已经对该文章点赞：通过查找 ArticleLikes 表中的记录，判断当前用户是否已点赞该文章。如果用户没有点赞，则插入一条新的点赞记录，记录点赞的文章 ID、用户 ID 和点赞时间。点赞后，重定向到文章详情页（view）展示最新的点赞信息。

```php
public function actionLike($id)
{
    // Ensure the article exists
    $this->findModel($id);

    // Check if the user has already liked the article
    $existingLike = ArticleLikes::find()->where([
        'article_id' => $id,
        'user_id' => Yii::$app->user->id,
    ])->one();

    if (!$existingLike) {
        $like = new ArticleLikes();
        $like->article_id = $id;
        $like->user_id = Yii::$app->user->id ?? null; // Assign user ID if logged in
        $like->created_at = date('Y-m-d H:i:s');
        $like->save();
    }

    return $this->redirect(['view', 'id' => $id]);
}
```

Views\Articles\index.php

GridView 是一个用于显示数据列表的 Yii2 小部件，它通过提供的数据提供器（$dataProvider）展示文章数据。filterModel 用于为表格提供搜索功能，通过 $searchModel 进行数据的过滤和搜索 yii\grid\SerialColumn：自动为每一行数据生成一个序号（从 1 开始）。id、title、created_at、updated_at：显示文章的 id、title（标题）、created_at（创建时间）和 updated_at（更新时间）字段。每个字段会自动从数据中提取并显示对应的值。Content 列：为每篇文章提供一个 "View Content" 按钮，点击按钮后会跳转到对应文章的详情页面（articles/view）。'label' => 'Content'：列标题为 "Content"。'format' => 'raw'：该列的数据是原生的 HTML 格式，而不是文本，允许包含按钮。'value'：该列的值是通过回调函数生成的。回调函数会为每篇文章生成一个 "View Content" 按钮，按钮的 onclick 事件将页面重定向到 articles/view 页面，显示该文章的详细内容。$data->id 获取当前文章的 id。

```php
<?= GridView::widget([
    'dataProvider' => $dataProvider,
    'filterModel' => $searchModel,
    'columns' => [
        ['class' => 'yii\grid\SerialColumn'],

        'id',
        'title',
        'created_at',
        'updated_at',

        // 显示内容按钮
        [
            'label' => 'Content',
            'format' => 'raw',
            'value' => function($data) {
                return Html::button('View Content', [
                    'class' => 'btn btn-primary',
                    'onclick' => 'window.location.href="' . Yii::$app->urlManager->createUrl(['articles/view', 'id' => $data->id]) .
                ]);
            },
        ],
```

Views\Articles\view.php

文章内容通过一个 iframe 标签显示，src 属性指向文章的内容 URL（$model->content_url）

```
<!-- 文章内容 -->
<div class="article-content">
    <h2>文章内容</h2>
    <div class="iframe-container" style="width: 100%; height: 600px; border: 1px solid #ddd; margin-bottom: 20px;">
        <iframe
            src="<?= $model->content_url ?>"
            style="width: 100%; height: 100%; border: none;"
            title="文章内容">
        </iframe>
    </div>
</div>
```

显示与文章相关的图片，如果有的话。每张图片以 img-thumbnail 类样式呈现。使用 foreach 循环遍历 $images 数组中的图片。如果没有相关图片，则显示 "暂无相关图片"。

```
    <!-- 显示图片 -->
<div class="article-images">
    <h3>相关图片</h3>
    <?php if (!empty($images)): ?>
        <div class="row">
            <?php foreach ($images as $image): ?>
                <div class="col-md-3 text-center">
                    <img src="<?= Html::encode($image->url) ?>" class="img-thumbnail" alt="文章图片" style="max-height: 150px;">
                </div>
            <?php endforeach; ?>
        </div>
    <?php else: ?>
        <p>暂无相关图片。</p>
    <?php endif; ?>
</div>
```

展示与文章相关的视频内容。每个视频使用 embed-responsive 类，以 16:9 的比例嵌入显示。如果没有相关视频，则显示 "暂无相关视频"。

```
<!-- 显示视频 -->
<div class="article-videos">
    <h3>相关视频</h3>
    <?php if (!empty($videos)): ?>
        <div class="row">
            <?php foreach ($videos as $video): ?>
                <div class="col-md-6">
                    <div class="embed-responsive embed-responsive-16by9">
                        <iframe
                            src="<?= Html::encode($video->url) ?>"
                            class="embed-responsive-item"
                            allowfullscreen>
                        </iframe>
                    </div>
                </div>
            <?php endforeach; ?>
        </div>
    <?php else: ?>
        <p>暂无相关视频。</p>
    <?php endif; ?>
</div>
```

通过 Html::a() 生成一个链接，用户点击后会执行一个 POST 请求，调用 like 操作并传递文章 ID（$model->id）。

data-method 设置为 post，确保该操作是通过 POST 请求完成的，避免通过 GET 请求进行点赞。

```
<!-- 点赞功能 -->
<div class="article-actions">
    <h3>操作</h3>
    <p>
        <?= Html::a("👍 点赞 ($likeCount)", ['like', 'id' => $model->id], [
            'class' => 'btn btn-success',
            'data-method' => 'post',
        ]) ?>
    </p>
</div>
```

显示文章的所有评论（如果有的话）。每个评论包括用户的 ID（$comment->user_id）、评论内容（$comment->content）以及评论的时间（$comment->created_at）。如果没有评论，则显示提示消息 "还没有评论，快来抢沙发吧！"用户可以在此评论区发表评论。使用 ActiveForm 渲染评论表单。表单包含一个文本区域字段，用户可以在其中输入评论内容。$newComment 是用于提交新评论的模型实例。当用户点击提交按钮后，表单数据会被提交到服务器。

```
<!-- 评论区域 -->
<div class="comments">
    <h3>评论</h3>
    <?php if (!empty($comments)): ?>
        <?php foreach ($comments as $comment): ?>
            <div class="comment">
                <strong>用户 <?= Html::encode($comment->user_id) ?>:</strong>
                <p><?= Html::encode($comment->content) ?></p>
                <p class="text-muted"><small>评论时间: <?= Html::encode($comment->created_at) ?></small></p>
            </div>
            <hr>
        <?php endforeach; ?>
    <?php else: ?>
        <p>还没有评论，快来抢沙发吧！</p>
    <?php endif; ?>
</div>

<!-- 评论表单 -->
<div class="comment-form">
    <h3>发表评论</h3>
    <?php $form = ActiveForm::begin(); ?>

    <?= $form->field($newComment, 'content')->textarea(['rows' => 4, 'placeholder' => '请输入您的评论...'])->label(false) ?>

    <div class="form-group">
        <?= Html::submitButton('提交', ['class' => 'btn btn-primary']) ?>
    </div>

    <?php ActiveForm::end(); ?>
</div>
```