搜索引擎指根据用户需求与一定算法,运用特定策略从互联网中检索出指定信息反馈给用户的一门检索技术。搜索引擎依托网络爬虫、网页处理、检索排序、自然语言处理等技术，实现网络信息获取、数据索引、检索、排序等功能。该项目通过实现搜索引擎基本功能，并为用户提供交互界面,构建了一套搜索引擎软件系统。项目产品四大核心功能——数据爬取、索引入库、用户检索、结果呈现，与商用级搜索引擎高度一致；项目模块基于主流技术设计实现，自动化程度高，可维护性强，易于部署，符合最佳实践标准。

项目架构层面,以知乎、博客园两大门户网站文章为数据源,针对搜索引擎爬取,处理、索引与持久化与用户检索服务功能，设计实现爬虫服务器、搜索服务器与应用服务器。爬虫服务器基于 python Scrapy 实现，在门户网站爬取数据，并将目标数据上报搜索服务器；搜索服务器以 ElasticSearch 为核心，包含 ElasticSearch-Head 控制面板与 Kibana 分析工具，另有 MySQL 关系型数据库，作为一个整体，完成数据的处理、索引与持久化，实现搜索后台逻辑，为应用服务器提供检索接口；应用服务器则通过 python Django 服务器与 Vue 前端 WEB 应用，以网页形式向用户提供搜索服务。下面，我们以一条互联网数据从被爬取到最终呈现给用户的旅程，进行项目功能的介绍。

首先，是数据爬取环节。爬虫使用 Undetected ChromeDriver 驱动与 Selenium 模拟浏览器进行操作，获取 cookie 后通过鉴权，再发起 HTTP 请求，获取目标数据，使用网站数据模型类对其格式化，最后通过 yield 送入数据流水线 Pipeline 进行后续操作。项目基于 python cv2 库，通过灰度图转化，实现知乎滑动拼图验证码离线自动识别破解功能。对于最为常见的两种爬虫工作方式——基于 HTML 解析与基于 AJAX 请求，项目均有涉及——爬取博客园网站时，通过分析网站代码，提取了获取对应文章详细信息的 AJAX 请求，爬虫直接通过该请求获取数据，效率大大提高；对于知乎网站，发现 AJAX 请求接口有加密处理，逆向难度较大，故爬虫通过分析 HTML 文档，使用 CSS 选择器定位并提取内容，节省大量时间。为应对网站反爬措施，项目爬虫在 settings.py 文件中规定限速，并采用随机可用 IP 与 FakeUserAgent 随机可用浏览器 UA，被识别出的概率大大降低，可进行长时间爬取。

其次，是索引入库环节。项目针对不同网站设计的数据模型类由同一个基类派生，需要实现 get_insert_sql 与 save_to_es 方法，并使用爬取的数据实例进行初始化。初始化完成后,通过 yield 送入 Pipeline。Pipeline 支持并行处理,编写 Pipeline 后,可在 settings.py 配置是否使用，灵活切换数据处理方式。项目 Pipeline 设计了 ElasticSearch 数据流处理器、MySQL 数据流处理器、数据流 JSON 文件生成器与图片批量下载器，分别满足不同需求。MySQL 数据流处理器中，调用传入数据模型实例的 get_insert_sql 方法，使用 python pymysql 驱动完成 MySQL 数据增量入库；ElasticSearch 数据流处理器调用传入数据模型示例的 save_to_es 方法，完成该条数据的处理、索引。对于索引的建立，项目在 elasticsearch_type.py 中以索引类的方式，规定了索引所需字段与 analyzer，首先调用 init 方法，在 ElasticSearch 中创建索引，在索引入库时，analyzer 会自动分析相关字段，完成分词等数据处理操作，再将数据入库。

然后，是用户检索环节。搜索服务器与应用服务器协同配合，实现用户检索的前后端逻辑，并对最终环节的 WEB 应用提供 RESTful API 数据接口。搜索服务器端，ElasticSearch 承担项目搜索核心功能，一方面，接收，处理，并存储来自 Pipeline 的数据模型示例，另一方面，对控制面板与分析工具提供数据接口，并支持高亮（Highlight）等开发者自定义的查询结果默认操作；ElasticSearch-Head 为 ES 控制面板，可供管理员查看、管理 ElasticSearch 的索引与数据；Kibana 分析工具对外提供 RESTful API 数据接口，负责接受来自应用服务器的用户请求，与 ElasticSearch 交互，获取检索数据，并响应应用服务器。在 ElasticSearch 、Kibana 实现后台搜索业务逻辑后,Django 应用服务器通过 urls 与 view 模块实现前端搜索业务逻辑——向搜索网页提供服务接口，根据用户自网页发送的搜索请

求，与 Kibana 交互，获取搜索结果，并响应用户。

最后，是结果呈现环节。该部分基于 B-S 架构，采用时下最为流行的前后端分离模式实现。前端为基于 Vue 3 构建的单页 WEB 应用；UI 使用 Element，美观大方，简洁明了，体验流畅；支持搜索输入提示、搜索历史、数据概览、分页等使用功能；参考主流搜索引擎，将搜索结果命中部分高亮处理，可通过点击跳转源页面，方便快捷。前端 WEB 应用通过 axios 库向应用服务器发起 AJAX 请求的功能，封装为两个 RESTful API——suggest 与 search_result，两者均为 GET 请求。两个接口均接受用户输入的关键词，suggest 返回搜索建议列表；search_result 接口返回搜索结果相关数据。用户在输入框输入搜索文本时，HTML 组件对用户操作进行消抖，通过事件绑定调用 suggest 接口，获取建议列表后，直接弹出建议窗口供用户选择；用户点击“搜索”按钮时，通过事件绑定调用 search_result 接口，在搜索结果区域，按关联度降序展示搜索结果。

至此，一条互联网数据从被爬取到最终呈现给用户的旅程，告一段落。海量数据也正是通过这种方式，在用户与搜索引擎的交互中，默默发挥着自身的潜在价值。项目通过运用 scrapy、MySQL、ElasticSearch、Django、Vue 等主流开源框架，融通前后端知识，打通数据爬取、索引入库、用户检索、结果呈现四大功能，实现网站增量爬取与快速接入，构建了可靠而强大、简单而易用、健壮而易于扩展的搜索引擎，是可谓一次搜索引擎系统设计与开发的最佳实践。项目团队成员也将在今后的学习实践中，继续探索、了解相关知识，运用理论与实践工具，在大数据中，发现更大价值。

项目报告到此结束，感谢您的聆听。