Week2

Responsible for back-end technology selection, specification of coding requirements and selection of good dependencies. Define the back-end package structure and create the initial back-end project, then upload it to github, initialise the front-end project and upload it to github.

The master branch was the front-end project and the main project was the back-end project. The database design was discussed and the database design conventions were standardised (see Database.docx for details). Identified the modules that needed to be done and implemented the database table design and links, and created the mysql database.

Week 3

Completed jira backend task assignment, configured email to send captcha, implemented registration and login.

Created the JWT tool class to generate JWTs based on user IDs and parsed JWTs to return user IDs.

Completed the registration interface (detecting whether the verification code is correct and determining whether the mailbox has been registered) and generated token and userId for the front-end.

Completed the image verification code acquisition interface and the login interface. Finished writing the proposal together and submitted it successfully.

Week 4

Create Book Information, Homebook entity class, Mybatis.

Completing the cross-domain problem on the front and back ends, Spring MVC provides a cross-domain solution.

Cross-domain is when a browser requests a resource from one domain from a web page in another domain, and the domain name, port and protocol are any different. In a front and back-end separation model, the domain names of the front and back ends are not the same, and cross-domain access problems can occur at this point.

Finish configuring the local cache with Caffeine. Caffeine is a Java 8 rewrite of the Google Guava cache and is a high-performance caching library that provides near-optimal hit rates.

Completing the configuration of the distributed cache Redis, the local cache has the advantage of fast access, but is unable to store large data. And when we deploy multiple service nodes in a cluster, or split the service later as the business grows, there is no way to share the cache and ensure consistency of the cached data. Locally cached data can also be lost on application restarts, which is not sufficient for data that needs to be persisted and can lead to excessive pressure on the database after a restart.

Completed interface to query books from homepage, first read from database but later read from local cache, speeding up reads.

Week 5

Completed the interface to query the books of the homepage, the first time reading from the database but later reading from the local cache to speed up the reading.

Added a list of hits, a list of newest books and a list of latest updated chapters. The hits list is sorted by the number of visitors, the latest books list is based on the latest list of book innovations and the latest updated chapters list is based on the latest list of chapter updates. I also finished basically all the Spring content and discussed with tutor what could be improved and where to go from here, and wrote a report.

Week 6

Completed the novel search module, you can search by keywords, you can search for books containing keywords, you can filter male and female books, you can also filter books by different categories and word counts, and you can also sort by different sorting methods, such as score scores and visits.

The interface to query categories is completed, allowing you to get specific fiction categories based on the direction of the work (boys or girls), while the caffeine cache is utilised here to query.

Week 7

Finished mybatis paging configuration, you can realize the normal paging

Query classification interface, access to the list and the latest novel list interface, novel details query interface, novel chapter name query interface.

Most of this data is added to the redis cache and a small portion is added to the caffeine cache.

Week 8

Added query the previous chapter chaperId interface, query the next chapter chaperId interface. Finished adding can collect books normally and cancel collection data, and can read novels according to history. Finished adding that you can get the chapter content and increase the number of clicks at the same time.

Modified some bugs, such as paging problems, and the use of redis cache without serialization.

Week 9

Started to implement the author module functionality, completed the interface for author registration, interface for changing author information, interface for getting author information.

Finished the interfaces to publish new books, update author's own book, and get his own book information.

Finished the interface that author can choose his own book to publish new chapters, the interface to update his own chapters and the interface to delete his own chapters.

Week 10

Started designing the fanfic module with a new interface to get information about other users.

Finished the interface that can choose to publish the fanfic under a book, can update their own fanfic information interface, can delete their own fanfic interface, can view the book's fanfic interface, can view all their own fanfic interface and according to the id of the interface to view fanfic information.

Optimization of the project, completed the use of Sentinel to implement the interface anti-scrubbing and flow-limiting and the use of the decorator pattern to solve the XSS attack on the form of passing parameters.