Terminology

Server

Term	Description
Runtime environment	The environment in which a program or application is executed. It's the hardware and software infrastructure that supports the running of a particular codebase in real time. Typically the runtime system will have some responsibility for setting up and managing the stack and heap, and may include features such as garbage collection, threads or other dynamic features built into the language.
npm (node package manager)	The package manager for the Node JavaScript platform **yarn** is a software packaging system developed in 2016 by Facebook for Node.js JavaScript runtime environment that provides speed, consistency, stability, and security as an alternative to npm (package manager).
Node.js	An open-source, cross-platform runtime environment that allows developers to create all kinds of server-side tools and applications in JavaScript. The runtime is intended for use outside of a browser context (i.e. running directly on a computer or server OS). As such, the environment omits browser-specific JavaScript APIs and adds support for more traditional OS APIs including HTTP and file system libraries.
Object-Relational Mapping (ORM)	A technique that lets you query and manipulate data from a database using an object-oriented paradigm. When talking about ORM, most people are referring to a *library* that implements the Object-Relational Mapping technique, hence the phrase "an ORM". An ORM library is a completely ordinary library written in your language of choice that encapsulates the code needed to manipulate the data, so you don't use SQL anymore; you interact directly with an object in the same language you're using.
Micro ORM	Basically a mapper that creates objects based on database query. How we are going to interact with database.
express	A Node Package Manager (npm) module (a.k.a. a package). it builds a rest api. The purpose of the express is that you don't have to repeat same code over and over again. Node.js is a low-level I/O mechanism which has an HTTP module. If you just use an HTTP module, a lot of work like parsing the payload, cookies, storing sessions (in memory or in Redis), selecting the right route pattern based on regular expressions will have to be re-implemented. With Express.js, it is just there for you to use.
apollo-server-express	The Apollo Server package for Express, the most popular Node.js web framework. It enables you to attach a GraphQL server to an existing Express server.

GraphQL	A query language for APIs and a runtime for fulfilling those queries with your existing data. GraphQL provides a complete and understandable description of the data in your API, gives clients the power to ask for exactly what they need and nothing more, makes it easier to evolve APIs over time, and enables powerful developer tools. GraphQL is not an ORM, because it doesn't understand the concept of DBs. It just gets the data from a "data source", which could be static, from a file, etc. Nor can it figure out how to get data once you point the source at it. You have to write resolver functions that tell the DB how to find the value of each field. It's also not a REST replacement, rather an alternative, and you can utilize both in the same project! GraphQL is an opinionated API spec where both the server and client buy into a schema format and querying format. Based on this, they can provide multiple advanced features, such as utilities for caching data, auto-generation of React Hooks based on operations, and optimistic mutations.
GraphQL resolver	A collection of functions that generate response for a GraphQL query. In simple terms, a resolver acts as a GraphQL query handler.
Express-session	An HTTP server-side framework used to create and manage a session middleware. A session will contain some unique data about that client to allow the server to keep track of the user's state. In session-based authentication, the user's state is stored in the server's memory or a database. When the client makes a login request to the server, the server will create a session and store it on the server-side. When the server responds to the client, it sends a cookie. This cookie will contain the session's unique id stored on the server, which will now be stored on the client. This cookie will be sent on every request to the server. Session can be stored in different places. In this project, I used Redis. The reason I am using this is because it is fast. This is important because in every request, I need to check if the user is logged in.
Redis	The open source, in-memory data store used by millions of developers as a database, cache, streaming engine, and message broker.
Middleware	A type of computer software that provides services to software applications beyond those available from the operating system. It can be described as "software glue".
Declaration files	Files that describe the shape of an existing JavaScript codebase to TypeScript. ex) yarn add -D @types/redis

Client

Term	Description
next.js	A React framework for developing single page Javascript applications. (https://snipcart.com/blog/next-js-vs-react)
chakra_ui	A simple, modular and accessible component library that gives you the building blocks you need to build your React applications.
urql	A lightweight, extensible GraphQL client for React. It is built to be easy to use, performant and functional, logical default behaviour and caching and easily extensible.
formik	Forms are a crucial component of React web applications. They allow users to directly input and submit data in components ranging from a login screen to a checkout page. Formik is a free and open source, lightweight library for ReactJS or React Native and addresses three key pain points of form creation: 1. How the form state is manipulated. 2. How form validation and error messages are handled. 3. How form submission is handled. The Formik library was written by Jared Palmer out of his frustration when building React forms, seeking to standardize the input components and flow of form submission. The idea was to keep things organized and in one place, simplifying testing, refactoring, and reasoning about your forms.

Server

Basic setting for server

- **npm init -y** # initiate a project (create package.json)
- **yarn add -D @types/node typescript** # add typescript and node and typescript behave each other so add dependency
- Add script in package.json
  - **ts-node** **#** typescript execution engine and REPL for Node.js. But this is slow, and in the production, usually they compile to javascript and run with node.
  - **nodemon** # listens for changes in files and reruns
  - **tsc-w** # take ts file and make js file in dist folder
- **yarn add -D nodemon** # How to install nodemon
- How to run the server app
  - **yarn watch** = tsc -w # in one terminal recompile type http://file3.instiz.net/data/cached_img/upload/2022/04/25/12/83334e61b1c7e855e01320a8728ccc6d.jpg script to javascript file
  - **yarn dev** = nodemon # in the other terminal re-execute the updated javascript file

Postgresql (in separate terminal)

- Set up postgresql
  - https://www.sqlshack.com/setting-up-a-postgresql-database-on-mac/
- **brew services start postgresql**

- **brew services stop postgresql**
- **psql -U postgres** # log in database with username postgres
- password: Juho19971106
- **CREATE DATABASE lireddit;** # create database

Mikro-orm
- **yarn add @mikro-orm/cli @mikro-orm/core @mikro-orm/migrations @mikro-orm/postgresql pg** //How to install micro ORM version 4 for postgresql
- command line (CLI):
  - **npx mikro-orm migration:create** # Create new migration with current schema diff
  - **npx mikro-orm migration:up** # Migrate up to the latest version
  - **npx mikro-orm migration:down** # Migrate one step down
  - **npx mikro-orm migration:list** # List all executed migrations
  - **npx mikro-orm migration:pending** # List all pending migrations
  - **npx mikro-orm migration:fresh** # Drop the database and migrate up to the latest version

Graphql
- **yarn add express apollo-server-express graphql type-graphql** # add few dependencies. Express is the server we are going to be using. apollo-server-express is a package that allow us to use graphql and create graphql endpoint very easily.
- **yarn add -D @types/express** # install typescript type for express.
- **yarn add reflect-metadata** # library that is used somewhere
- **yarn add argon2** # password hashing https://github.com/ranisalt/node-argon2

Connect Redis Middleware
- Set up redis
  - **brew install redis** # install redis
  - **redis-server** # start the redis server manually
- **yarn add redis connect-redis express-session** # add libraries
- **yarn add -D @types/redis @ types/express-session @tpyes/connect-redis** # install typescript type
- Checking cookies in Chrome browser
  - Go to inspect
  - In the network tab, can see all the requests that will happen
  - In the application tab, there is a cookies tab. At there can see cookies

Client

Next.js with chakra_ui
- **yarn create next-app --example with-chakra-ui <name of the folder>** # bootstrap the example code of next.js frontend app with chakra_ui library
- How to run the client app
  - **yarn dev** = next # execute the next.js app
- **yarn add formik** # add formic