Software Design Document

1. Introduction

1.1 Purpose

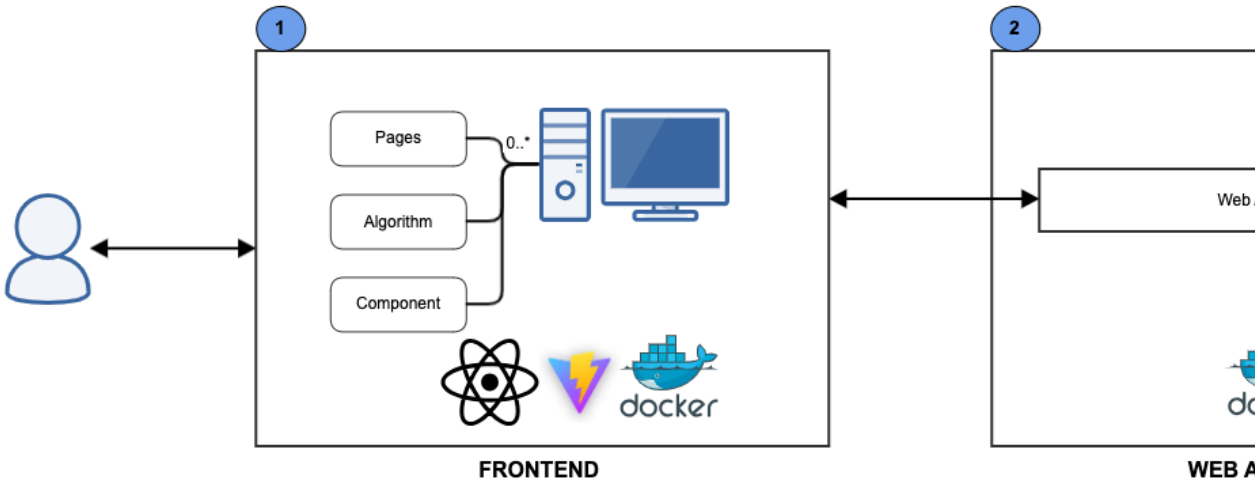
This document describes the design of Algo-Shadow web application.

1.2 Scope

This design document is for version V2 of Algo-Shadow, which includes high level architecture, modules, and UML diagrams. Since this application uses existing frameworks and libraries (SpringBoot, Vite, and Reacts), this design document excludes framework classes.

2. High Level Architecture

This application resembles a Client-Server-Database architecture, which comprises of 3 major components, including a web application, a web server, and a database. The client communicates with the server via REST APIs to then perform standard database functions. In addition, we choose a thick-client application that carries out algorithmic computations and animations to reduce load to our server and achieve a faster performance.



3. Front-end Application design

3.1. UI design

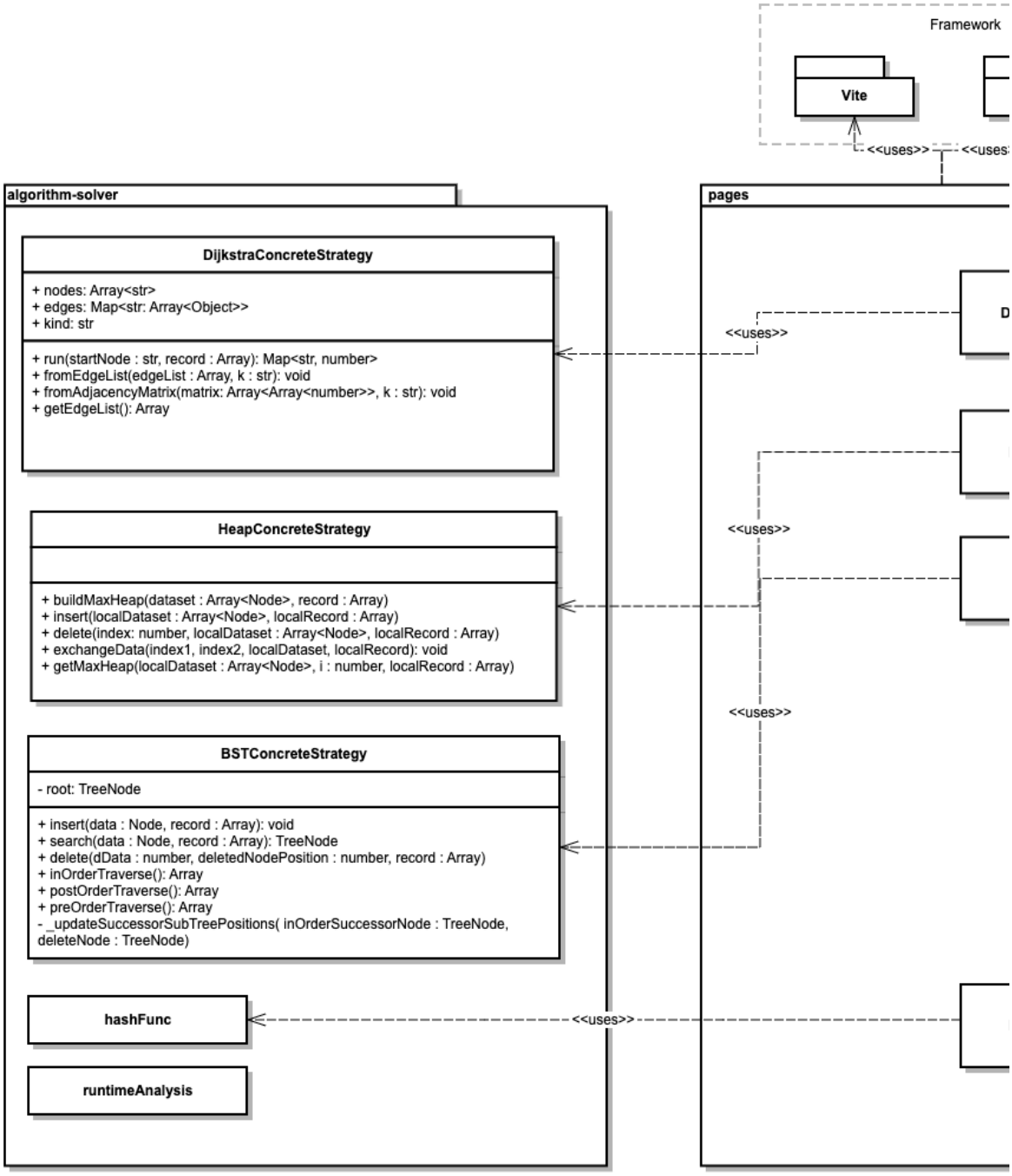
Please refer to Figma documents here https://www.figma.com/file/pULQC1OwkhB9GCFttLYN7d/Algo-Shadow?type=design&node-id=0-1&mode=design&t=8QuIa69w4l0Y8FAH-0

3.2. Client-side modules

On client-side, we have different modules handle different responsibilities, as described below:

- The `algorithm-solver` module consists of different algorithmic logic implementations, such as Dijkstra, Heap, etc. This module is responsible for running the algorithm, calculating the run-time, and returning the result for rendering.

- The `renderer` module consists of different implementations for visuals and graph animation using `d3` and `svg`. Depends on the type of algorithm (ie. Tree or DiGraph), different renderer is used.
- The `barchart-analyze` module is responsible for creating charts from experiment data retrieve from client local storage



3.3. Page components

Beside modules described above, we'll have stateless components implemented with React, as described below:

- `SaveInputToLocalStorage:` should save user experiment data, including type of operations, input data, output returned by `algorithm-solver`, and total runtime calculated by `RuntimeAnalysis` component

- `AuthGuard`: determine if the users is logged in or not. If not logged in, the user will not be redirected to the Algorithm pages
- `TableCreater`: renders a table on the UI with data retrieved from client's local storage

3.4. Local Storage

Past users experiment data will be stored in Local Storage client-side.

4. Back-end Server application design

4.1. AuthenticationService module

Authentication is handled with a different service, operating independently from Algo-Shadow application. Authentication service module and packages are created by hicsail in the repository https://github.com/hicsail/authentication-service. We will do our own hosting. Communication between the Algo-Shadow web application and Authentication service is done via a few Authentication Service APIs listed below

API Name	Description	Parameters
POST v1/auth/signup	The user is able to sign up	`email`: string `password`: string `lastName`: string `firstName`: string `buID`: string `role`: number
POST /auth/login	The user is able to sign in	`email`: string `password`: string

4.2. Authentication Database Detail

Database Engine: PostgreSQL

Host: <check with dev team>