Technical Design Document

Requirements

Functional Requirements:

- Web Based system
- A user can create an account
- A user can fund their account
- A user can transfer funds to another user's account
- A user can withdraw funds from their account.

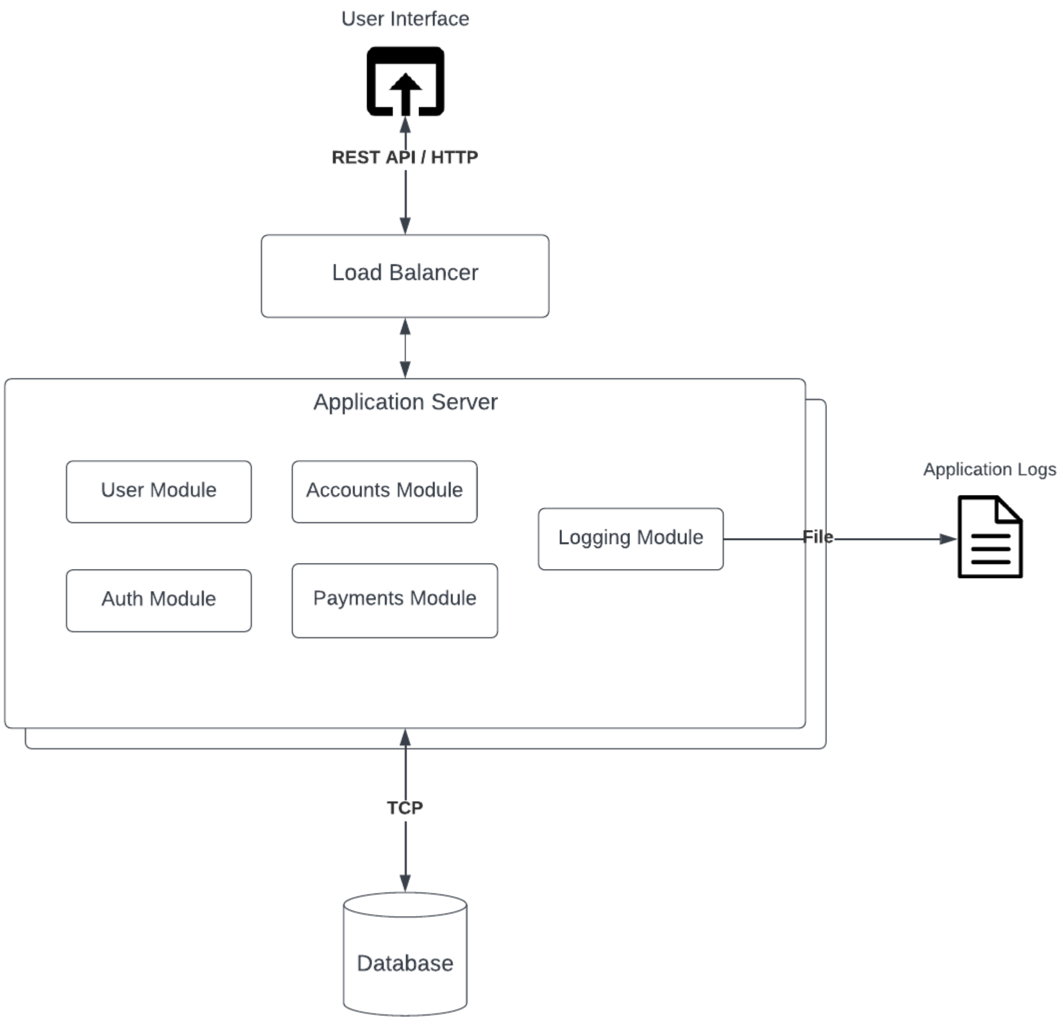
Non Functional Requirements:

- Full authentication system doesn't need to be built, a faux token based authentication will suffice.
- Project should be implemented using Node JS.
- MySQL is to be used for the database.

Components

S/N	Component	Description
1	Load balancer	Distributes incoming requests across server pools.
2	Application Server	An HTTP server.
3	User Module	Performs CRUD operations on users.
4	Accounts Module	Manages the account operations for users.
5	Logging Module	Writes Logs to the console, file, and database.
6	Auth Module	Handles authentication of the users.
7	Payments Module	Handles account transactions

Architecture



Notes

The following should be noted for the purposes of this exercise:

- Monolithic architecture was used because it was the fastest working solution I could build due to the time constraints.
- The account number generation could be more elaborate, but it works. 10,000 unique account numbers are seeded into a table.
- A Load balancer will not actually be implemented, but it is a proper representation of what is expected in production.
- For additional security, the request payloads can also be RSA encrypted (though not implemented here).
- Logging was not implemented due to time constraints.
- See repo on how to run and test project: https://github.com/Dr-Drake/payment-solution