HW#8 Subscription Patch

- Frontend: src/index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import {
  ApolloClient, InMemoryCache, ApolloProvider,
  split, HttpLink
} from '@apollo/client';
import { getMainDefinition } from
  '@apollo/client/utilities';
import { GraphQLWsLink } from
  '@apollo/client/link/subscriptions';
import { createClient } from 'graphql-ws';

import { ChatProvider } from
  "./containers/hooks/useChat"
import App from "./containers/App";
import reportWebVitals from "./reportWebVitals";

const httpLink = new HttpLink({
  uri: 'http://localhost:5001/graphql'
});
const wsLink = new GraphQLWsLink(createClient({
  url: 'ws://localhost:5001/subscriptions',
  options: {
    lazy: true,
  },
}));
```

```
const splitLink = split(
  ({ query }) => {
    const definition = getMainDefinition(query);
    return (
      definition.kind === 'OperationDefinition' &&
      definition.operation === 'subscription'
    );
  },
  wsLink,
  httpLink,
);

const client = new ApolloClient({
  link: splitLink,
  cache: new InMemoryCache(),
});
const root =
ReactDOM.createRoot(document.getElementById("root")
);
root.render(
  <React.StrictMode>
    <ApolloProvider client={client}>
      <ChatProvider><App /></ChatProvider>
    </ApolloProvider>
  </React.StrictMode>
);
reportWebVitals();
```

HW#8 Subscription Patch

- Backend: src/server.js
- yarn add ws

```
import * as fs from 'fs'
import { createServer } from 'node:http'
import { WebSocketServer } from 'ws'
import { createPubSub, createSchema,
createYoga } from 'graphql-yoga'
import { useServer } from 'graphql-ws/lib/
use/ws'
import ChatBoxModel from './models/chatbox'
import Query from './resolvers/Query';
import Mutation from './resolvers/Mutation';
import Subscription from './resolvers/
Subscription';
import ChatBox from './resolvers/ChatBox'

const pubsub = createPubSub();
```

```
const yoga = createYoga({
  schema: createSchema({
    typeDefs: fs.readFileSync(
      './src/schema.graphql',
      'utf-8'
    ),
    resolvers: {
      Query,
      Mutation,
      Subscription,
      ChatBox,
    },
  }),
  context: {
    ChatBoxModel,
    pubsub,
  },
  graphiql: {
    subscriptionsProtocol: 'WS',
  }
});
const httpServer = createServer(yoga)
const wsServer = new WebSocketServer({
  server: httpServer,
  path: yoga.graphqlEndpoint,
})
```

```
useServer(
  {
    execute: (args) => args.rootValue.execute(args),
    subscribe: (args) => args.rootValue.subscribe(args),
    onSubscribe: async (ctx, msg) => {
      const { schema, execute, subscribe, contextFactory, parse, validate } =
        yoga.getEnveloped({
          ...ctx,
          req: ctx.extra.request,
          socket: ctx.extra.socket,
          params: msg.payload
        })

      const args = {
        schema,
        operationName: msg.payload.operationName,
        document: parse(msg.payload.query),
        variableValues: msg.payload.variables,
        contextValue: await contextFactory(),
        rootValue: {
          execute,
          subscribe
        }
      }
      const errors = validate(args.schema, args.document)
      if (errors.length) return errors
      return args
    },
  },
  wsServer,
)
export default httpServer;
```