Project 2 – Readme (IFT3225)

Web Application: Word and Definition Game with REST API

a) Identity and Role

- **Name:** Ibrahim Charara
- **Student ID:** 20207667
- **Role:** Sole developer. I designed and implemented the entire backend (Node.js + MySQL + REST API) and frontend (HTML/CSS/JS) components of the project.

b) Hosting and Links

The application is hosted on Render.com because Node.js servers (such as Express apps listening on custom ports like 3000) are not supported by DIRO's www-ens infrastructure. Multiple solutions were attempted (including PHP proxy and CGI), but none allowed a clean exposure of a running server. Therefore, the fullstack application was deployed to Render.

- **Base URL:** `https://cross-word.onrender.com`

Main HTML Pages:

- `https://cross-word.onrender.com/jeu/word`
  Main word guessing game where the player must identify a hidden word letter by letter. Hints are revealed progressively and suggestions are available at a cost. The following frontend route formats are supported:
  - `/jeu/word`
  - `/jeu/word/lang`
  - `/jeu/word/lang/time/hint`

  Each version loads the same page with default or custom parameters passed via the URL path. The order of parameters is required.
- `https://cross-word.onrender.com/def.html`
  Game mode where users provide definitions for a given word under time constraints. Two launch formats are available:
  - `/def.html`
  - `/jeu/def/lang/time`

  The second format allows the language and duration to be pre-specified through the path.
- `https://cross-word.onrender.com/dump.html`
  Displays the database content (players, words, definitions) as a paginated and searchable table using DataTables.
  - `/api/dump` — returns 10 rows default

  - `/api/dump/:step` — returns paginated data by number of rows

  These routes are used by the frontend logic inside `dump.js` to populate the interface.

- `https://cross-word.onrender.com/doc.html`
  Interactive documentation for all available REST API endpoints. Includes live test forms for sending GET, POST, and PUT requests.

c) External Libraries and Tools

- **jQuery** – AJAX calls and DOM manipulation.
- **DataTables.js** – Dynamic table rendering with search/sort.
- **mysql2** – MySQL driver for Node.js.
- **Express.js** – Web server and routing layer.
- **bcryptjs** – Password hashing utility on the backend.
- **Render.com** – Free hosting platform for the full-stack deployment.

d) Key Features

- Interactive word guessing game with timer, automatic hint injection, and score tracking.
- Definition management: add, filter, and validate new entries.
- Player system: login/logout with persistent scores stored in MySQL.
- Fully functional REST API used by the frontend and testable via `curl` or forms.

e) Delivered Files

Frontend

- HTML pages: `jeu.html`, `def.html`, `dump.html`, `doc.html`
- JavaScript files: `word.js`, `def.js`, `login.js`
- CSS stylesheets: `style.css`, `doc.css`, `dump.css`

Backend

- `backend.js` – Express server handling all REST routes, game logic, and SQL queries.
- MySQL database – Tables for players, words, definitions, and scores with secure connections via `mysql2`.

The entire application was hand-coded without using frontend frameworks. All components are original except the external libraries listed above. Interfaces were tested via both browser and curl requests.