Wordingo

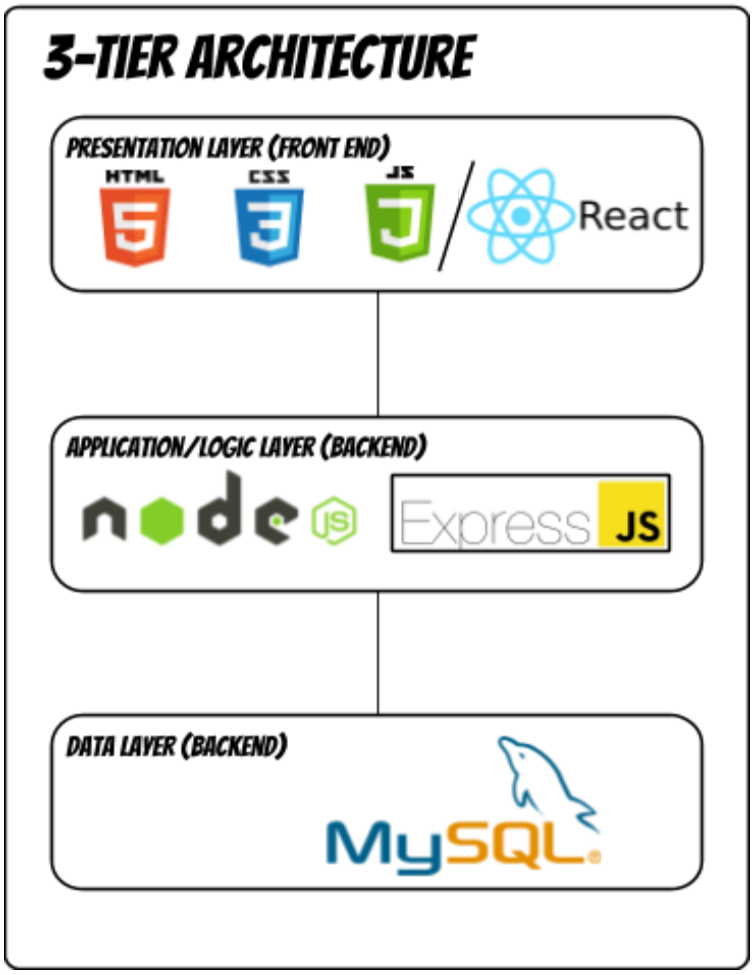
CS157A - Team 37
Instructor: Mike Wu

Xiaohan Sun
Andrew Wilson
Alexis Huerta

Project Overview

Wordingo is a user-defined dictionary application. This application will allow users to register accounts, search a MySQL database for word definitions, and add or edit their own words in the database. Search methods will include browsing alphabetically and via specific word lookup. Wordingo will employ a clean and easy to navigate GUI, implemented in javascript. GUI elements will include a search bar and a navigation sidebar.

System Environment



Stakeholders are specifically people who work in a field that has a rapidly and dynamically evolving jargon, such as medical professionals and software engineers. This is application is important and relevant because someone who works in such a field will often be confronted with terminology that they are not familiar with, and maybe in the position to coin a new word for a process or procedure that they originate. By allowing users to create their own words, our application will, therefore, have a high degree of applicability for these professions. Specifically, this application is intended to be deployed within an organization (such as a large tech company for example) to keep track of internal terminology.

Functional Requirements

High-level Requirement	Priority	Detail Leveled Requirements
Create Account	1	User enters a username and email.
		Username or email taken; prompt user to try again.
	2	User is prompted to create a password with at least 8 characters. Password is valid; user is allowed to continue.
		Password does not meet requirements. Prompt user to try again.
Login	3	User enters their email and password. If email and password are valid, user can press the “Sign In” button and enter the main page and access their account.
		If email or password is invalid; notify the user with an error message : “Invalid Credentials” and prompts the user to re-enter their information.
Search by Word	4	User searches a word in a search bar. If the word exists, show all definitions.
	5	If the word does not exist; notify the user with an error message: “Word does not exist.” and prompt user to re-enter another word.
Search by Alphabetical Browsing	6	User clicks a letter, and the dictionary shows all words starting with the letter alphabetically.
Submit New Words	7	User clicks the “Create” button which redirects them to create their own words with definitions and submit them to the dictionary.
Add a Description to an Existing Word	8	User adds a description to an existing word found in the dictionary.
Edit Their Own Word Definition	9	User chooses a word with a definition they have written and edits the definition.
Remove Words	10	User clicks a “Remove” button which redirects them to choose a word to delete. Notify the user with the message: “ Do you want to delete this word?”. If the User clicks the “Delete” button, the word is deleted from the dictionary; if the User clicks the “Cancel” button which redirects the user return to the previous page.
Upvote /Downvote Words	11	User clicks a button to upvote or downvote a certain definition for a word.

Non-functional Issues

1. GUI: The gui elements used will encompass a sidebar, a search bar, hyperlinks, and edit/delete word buttons.

GUI Element	**Details of function**
Sidebar	Navigate to add word page and alphabetical lookup page
Search bar	Enter a specific word and search database for it's definition
Hyperlink	Click on words when browsing database to view its definition
Edit button	Allows user to navigate to edit the description of a word that they created
Delete button	Allows user to delete a word that they created

2. Methods used to develop the GUI will include JavaScript, HTML, CSS and React. The Justinmind software will be used to prototype our interface.
3. Non-functional Requirements:Availability, Performance, Security, and Usability.

Availability	The system should have a high index of availability. The system should only be down for repairs and updates once every three months.
Performance	The system should respond to any user action (hyperlink selection, search, ect) in less than 5 seconds time.
Security	All user accounts should be secured with a password, and linked to an email address. User passwords should be enforced to be secure character strings.
Usability	The system should allow a naive user to understand how to search for words, and add/edit/delete words by reading page content without referring to additional documentation