**Converse about the Verse – Software Documentation**

**Shira Taitelbaum, Yonadav Leibowitz, Oriya Yehudai**



**Data:**

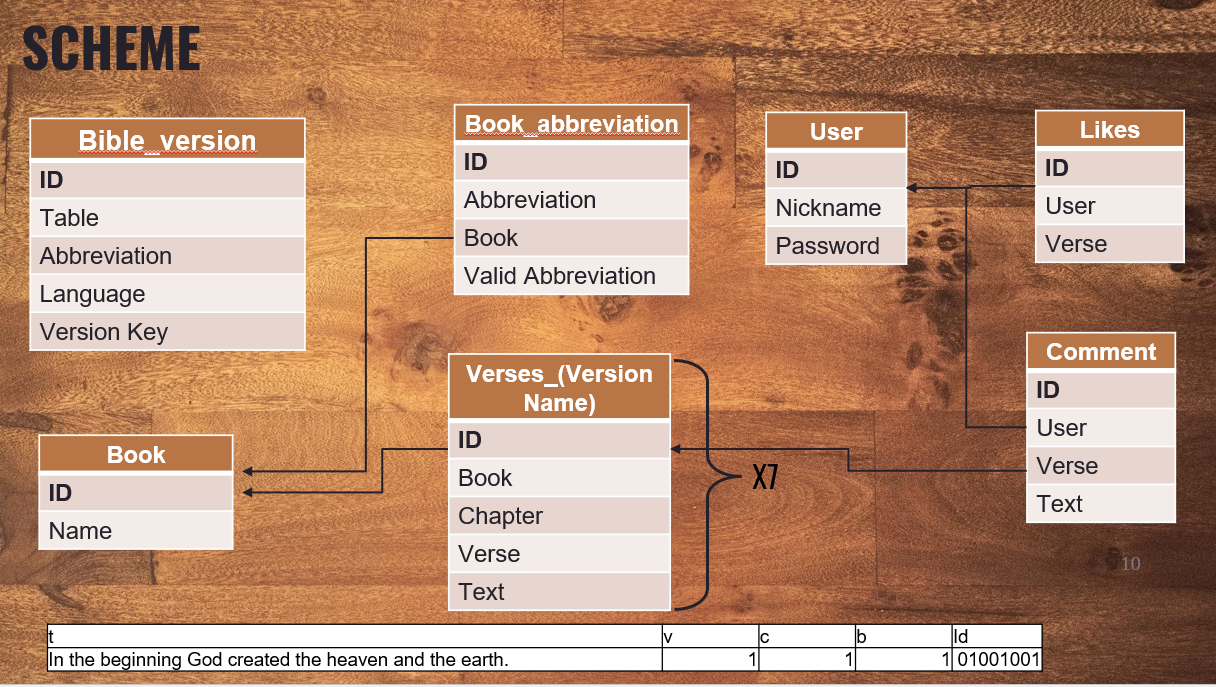
Our endeavor, at its core, is a social network designed to facilitate connections among individuals who are passionate about the Bible. The platform allows users to connect with one another, share and discover biblical verses, and engage in meaningful discussions and reflections.

To provide a rich and diverse selection of biblical content, we sourced our initial dataset from the popular online platform, Kaggle. Specifically, we accessed the data via the following link: https://www.kaggle.com/datasets/oswinrh/bible. This dataset comprises seven different English translations of the Bible, as well as a mapping table that matches each book id to book title.

Upon obtaining the data, we faced the task of preparing it for use on our platform. The original format of the dataset was in csv, and to import it into our system, we utilized the "benchmark" desktop application of "mysql". Through this process, we were able to export the data to a MySQL server, where it is now readily available for use on our platform.

We subsequently introduced an additional component to our database: a table that stores the usernames and passwords of our users, as well as their associated comments. Additionally, we created another table that records the specific verse identifiers that have been liked by each user, and a separate table for the comments that users have made on individual verses.

**Scheme:**



**Queries:**

def get\_all\_translations\_names() :

    cursor.execute('SELECT version FROM bible\_version\_key WHERE version IS NOT NULL;'.format())

This query is trying to get the name of all translations that are available in our database.

def size\_likes() :

    cursor.execute("SELECT COUNT(\*) FROM likes")

The query is selecting the count of the rows of a table called "likes" using the SQL function "COUNT(\*)". The query is trying to get the number of rows in the table "likes" which represents the number of likes in the system.

def get\_all\_book\_names() :

    cursor.execute('SELECT n FROM key\_english WHERE n IS NOT NULL;'.format())

The query is selecting the column "n" from a table called "key\_english" where the column "n" (name) is not null. This query is trying to get the name of all books that are available in the table and not null.

def book\_id\_to\_title(bookId) :

    cursor.execute('SELECT n FROM key\_english WHERE b = {} AND n IS NOT NULL;'.format(bookId))

The query is selecting the column "n" from a table called "key\_english" where the column "b" is equal to the value of "bookId" passed into the function and the column "n" is not null. This query is trying to get the title of the book by providing the book id.

def liked\_verses(\_user, table) :

    cursor.execute('SELECT \* FROM {} WHERE id IN (SELECT id\_verse FROM likes WHERE user="{}");'.format(table, \_user))

The subquery selects the column "id\_verse" from the table "likes" where the column "user" is equal to the value of the parameter "\_user" passed into the function. The outer query is trying to get all the verses that are liked by a specific user.

def search(table, word) :

    cursor.execute('SELECT \* FROM {} WHERE t LIKE "%{}%";'.format(table, word))

The query is selecting all the rows from the table passed in as the parameter "table" where the column "t" contains the value of the parameter "word" passed into the function. The query is using the "LIKE" operator and the wildcard "%" to match any string that contains the word passed in. This query is trying to search for all the verses that contain a specific word in the text.

def table\_book(table, book\_no) :

    if not book\_no.isnumeric() :

        book\_no = str(book\_title\_to\_id(book\_no))

    cursor.execute('SELECT \* FROM {} WHERE b = {} AND t IS NOT NULL;'.format(table, book\_no))

The query is selecting all the rows from the table passed in as the parameter "table" where the column "b" is equal to the value of the parameter "book\_no" passed into the function and the column "t" is not null. This query is trying to get all the verses of a specific book, where the text of the verse is not null.

def liked\_verses\_by\_book(table, book\_no) :

    book\_title = book\_id\_to\_title(book\_no)

    cursor.execute('SELECT \* FROM {} WHERE b={} AND id IN (SELECT id\_verse FROM likes);'.format(table, book\_no))

The query is selecting all the rows from the table passed in as the parameter "table" where the column "b" is equal to the value of the parameter "book\_no" passed into the function and the column "id" is in a set of values that come from a subquery. The subquery selects the column "id\_verse" from the table "likes". The outer query is trying to get all the verses of a specific book that have been liked by any user.

def num\_likes\_by\_book(table) :

    cursor.execute(

        'SELECT n, COUNT(\*) FROM (({} AS bible JOIN likes ON bible.id = likes.id\_verse ) JOIN key\_english ON bible.b = key\_english.b ) GROUP BY key\_english.n  ;'.format(

            table))

The query is selecting the column "n" and the count of rows from the result of a join of three tables: the table passed as the parameter "table", the "likes" table, and the "key\_english" table. The join is performed such that the column "id" of the table passed as parameter is equal to the column "id\_verse" of the "likes" table and the column "b" of the table passed as parameter is equal to the column "b" of the "key\_english" table. The query then groups the results by the column "n" of the "key\_english" table. The query is trying to get the number of likes for each book of the bible.

def table\_book\_chapter(table, book\_no, chapter\_no) :

    if table[0 : 2] != 't\_' :

        table = str(translation\_name\_to\_id(table))

    if not book\_no.isnumeric() :

        book\_no = str(book\_title\_to\_id(book\_no))

    # Query the MySQL database and generate the HTML table

    cursor.execute('SELECT \* FROM {} WHERE b = {} AND c = {} AND t IS NOT NULL;'.format(table, book\_no, chapter\_no))

    rows = cursor.fetchall()

def userliked(table, book\_no, chapter\_no, verse\_no) :

    cursor.execute(

        'SELECT \* FROM {} WHERE b = {} AND c = {} AND v = {} AND t IS NOT NULL;'.format(table, book\_no, chapter\_no,

                                                                                        verse\_no))

This query selects all rows from the specified bible version where the values of the columns "b", "c", "v" match the provided "book\_no", "chapter\_no", and "verse\_no" respectively and the value of the column "t" is not null.

@app.route('/signin', methods=['POST'])

def login\_action() :

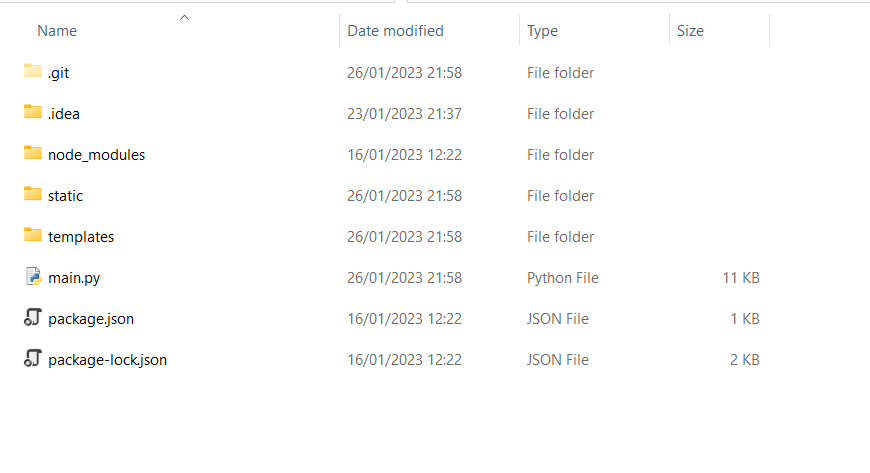
    username = request.form['username']

    password = request.form['password']

    cursor.execute('SELECT \* FROM users\_passwords WHERE id = "{}" AND pass = "{}";'.format(username, password))

This query selects all rows from the table 'users\_passwords' where the value of the column 'id' matches the provided username and the value of the column 'pass' matches the provided password.

**Code structure:**



The **main.py** file serves as the central hub for our web application, utilizing the Flask framework to efficiently handle both GET and POST requests from users. It communicates with our database, executing any necessary queries, and subsequently serves HTML pages to the user. The **node\_modules** directory holds a Bootstrap module, which provides a sleek and visually pleasing design for our HTML templates. The **templates** directory stores the HTML templates that are utilized by our server to dynamically load data and serve it to the client. Finally, the **static** directory contains images and CSS files that are imported by the HTML templates to enhance the overall aesthetic appeal of the application.