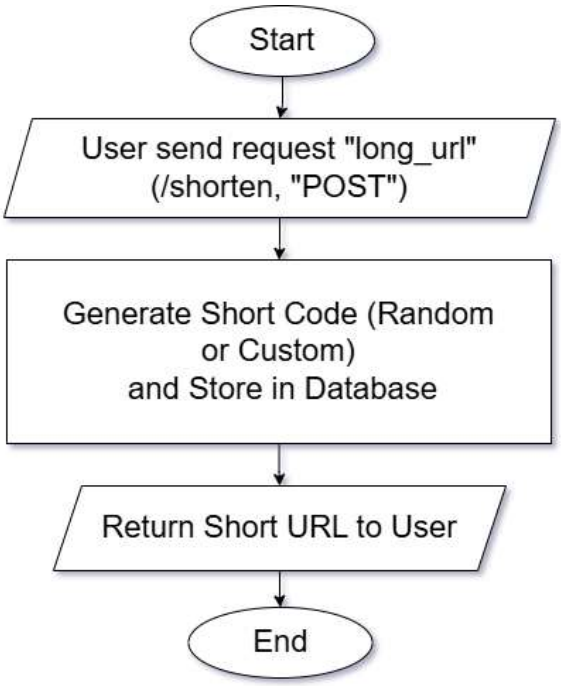
Project 1 - URL Shortener using Go

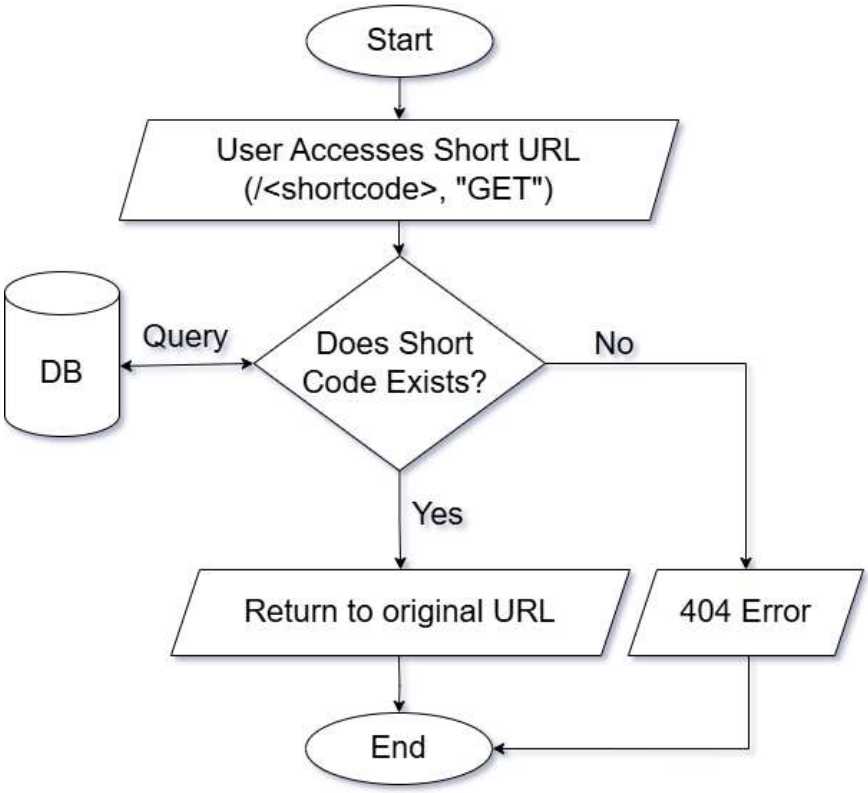
Problem of the Project

1. Input: long url
2. Output: shortened url route

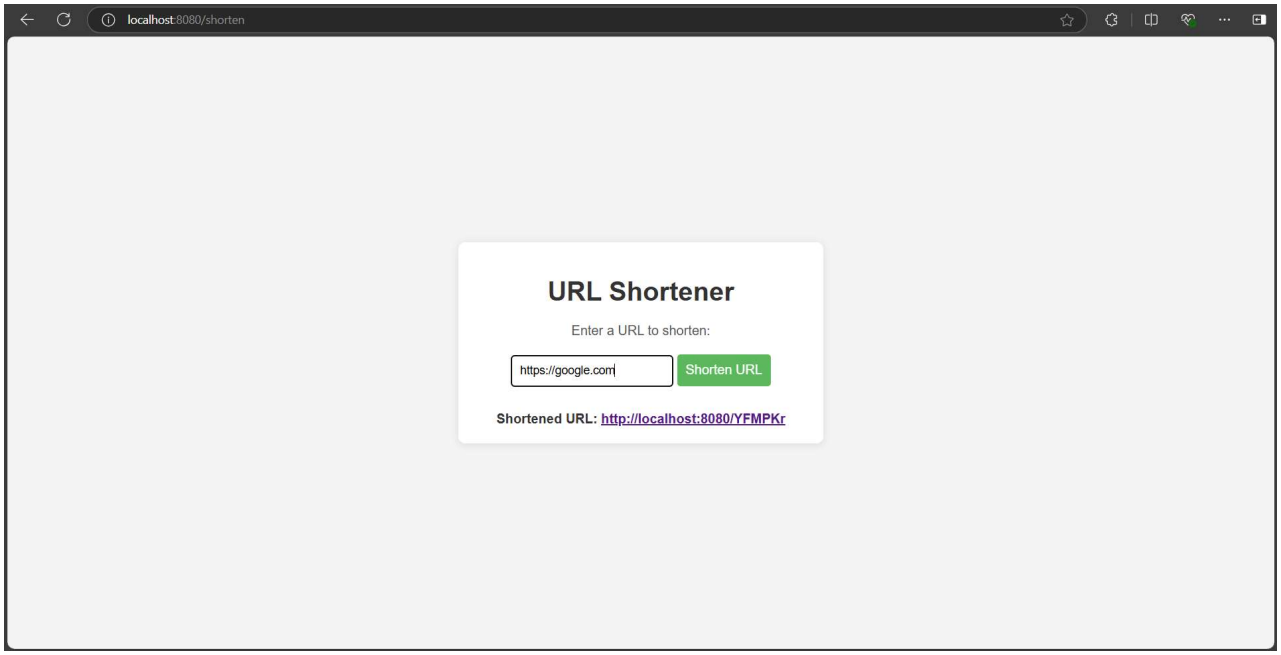
Tech Stack: HTML, CSS, Go, MongoDB Atlas

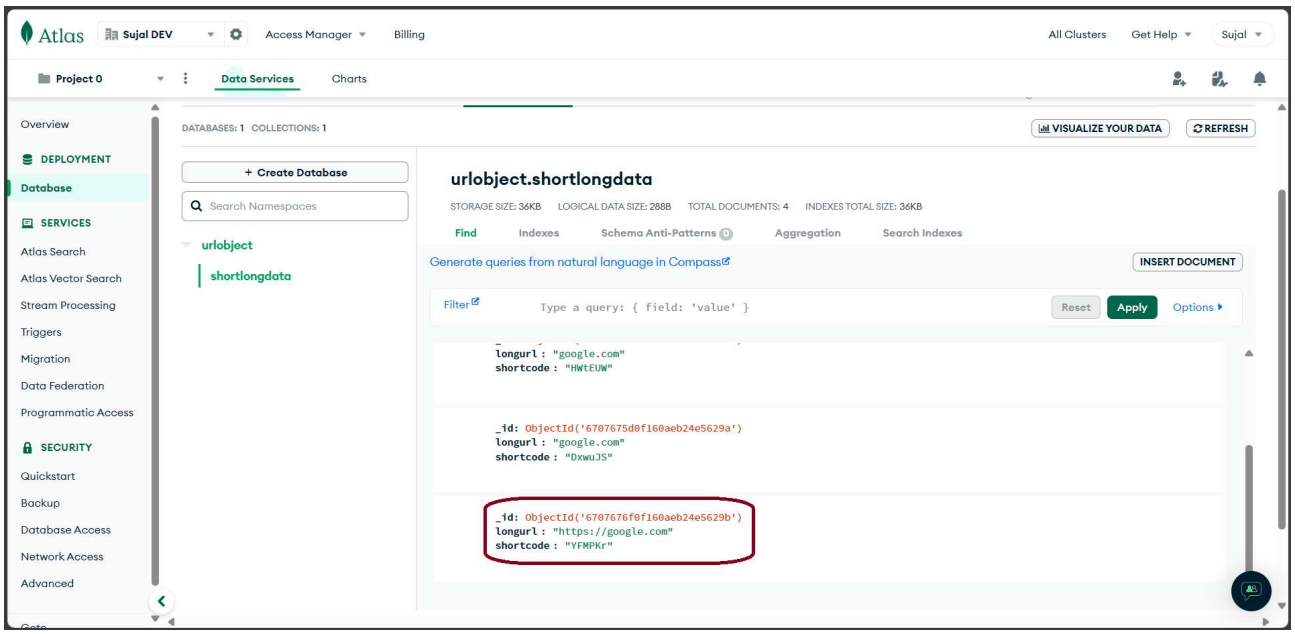
Flowchart





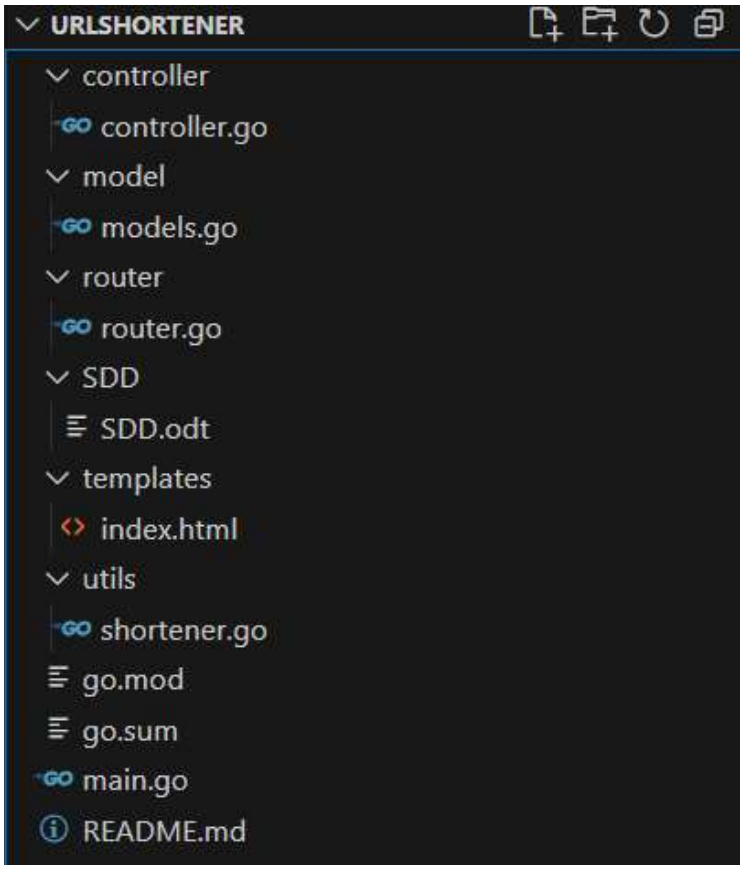
OUTPUT


localhost:8080/shorten
URL Shortener
Enter a URL to shorten:
https://google.com
Shorten URL
Shortened URL: http://localhost:8080/YFMPKr


Atlas
Sujal DEV
Access Manager
Billing
All Clusters
Get Help
Sujal
Project 0
Data Services
Charts
Overview
DEPLOYMENT
Database
SERVICES
Atlas Search
Atlas Vector Search
Stream Processing
Triggers
Migration
Data Federation
Programmatic Access
SECURITY
Quickstart
Backup
Database Access
Network Access
Advanced
DATABASES: 1 COLLECTIONS: 1
VISUALIZE YOUR DATA
REFRESH
+ Create Database
Search Namespaces
urlobject
shortlongdata
urlobject.shortlongdata
STORAGE SIZE: 36KB LOGICAL DATA SIZE: 288B TOTAL DOCUMENTS: 4 INDEXES TOTAL SIZE: 36KB
Find
Indexes
Schema Anti-Patterns
Aggregation
Search Indexes
Generate queries from natural language in Compass
INSERT DOCUMENT
Filter
Type a query: { field: 'value' }
Reset
Apply
Options
longurl : "google.com"
shortcode : "HWtEUW"
_id: ObjectId('6707675d0f160aeb24e5629a')
longurl : "google.com"
shortcode : "DxwuJS"
_id: ObjectId('6707676f0f160aeb24e5629b')
longurl : "https://google.com"
shortcode : "YFMPKr"

Go Implementation

Directory structure (MVC- Model View Controller Approach)



Controller: It is the most important folder which handles all the important functions of this project. Such as connecting to the MongoDB Atlas Servers, shortening URLs, redirecting back shortened URLs to original URLs,etc.

Model: It is where the json/bson structure of the URL object is defined which consists of two components - longurl and shortcode

Router: Corresponding controller function calls according to the routes are written inside this file

SDD: (or) Software Design Document is the document which consists of the detail documentaion and functions of this project "URL Shortener using Go"

Templates: it is the directory where html files are saved in order to serve when the HOME page is needed

Utils: (or) Utilities is the directory where shortener.go is saved which is responsible for the function of randomly generating the short code URL by using a random seed function from math pkg

Main.go: It is the main file where the program starts and using `http.ListenAndServe()` listens to a specific port(i.e. here 8080) and gives signal to the router to start

Controller.go Working Structure

1. init() Function

```go
// Connect with MongoDB
func init() {
    // Client options
    clientOption := options.Client().ApplyURI(connectionString)

    // Connect to MongoDB
    client, err := mongo.Connect(context.TODO(), clientOption)
    if err != nil {
        log.Fatal(err)
    }
    fmt.Println("MongoDB connection success")

    collection = client.Database(dbName).Collection(colName)

    // Collection instance
    fmt.Println("Collection instance is ready")
}
```

Purpose: responsible for connecting to the database and using database name(**dbName**) and collection name(**colName**) and giving out collection variable so that operations can be performed on **colName** collection

2. inserShortenUrlRecord() Function

```go
// MongoDB helpers - file
func insertShortenUrlRecord(urlObj model.UrlObject) {
    _, err := collection.InsertOne(context.TODO(), urlObj)
    if err != nil {
        log.Fatal(err)
    }
}
```

Purpose: pushes an URL object (which contains shortcode and corresponding longurl) to the MongoDB database collection

3. checkShortenUrl() Function

```go
// MongoDB helpers - file
func checkShortenUrl(shortCode string) (model.UrlObject, error) {
    var urlObj model.UrlObject
    filter := bson.M{"shortcode": shortCode}
    err := collection.FindOne(context.TODO(), filter).Decode(&urlObj)
    return urlObj, err
}
```

Purpose: checks whether the newly created shortcode already exists in Database for eliminating coinciding situations. **bson** is being passed with the shortcode and this **filter** is then searched in the collection

4. GetHomePage() Function

```go
// Controller - file
func GetHomePage(w http.ResponseWriter, r *http.Request) {
    // Parse the HTML template
    tmpl, err := template.ParseFiles("templates/index.html")
    if err != nil {
        http.Error(w, "Unable to load template", http.StatusInternalServerError)
        return
    }

    // Execute the template and write to the response
    err = tmpl.Execute(w, nil)
    if err != nil {
        http.Error(w, "Unable to execute template", http.StatusInternalServerError)
    }
}
```

Purpose: It parses the **"index.html"** file from **"templates/index.html"** directory and loads it into **tmpl** and then **tmpl.Execute()** loads the file in the localhost **"/"** route

5. ReturnOriginalUrl() Function

```go
func ReturnOriginalUrl(w http.ResponseWriter, r *http.Request) {
    shortCode := mux.Vars(r)["shortcode"] // Get the shortcode from the URL

    // Check if the shortcode exists
    urlObj, err := checkShortenUrl(shortCode)
    if err != nil {
        http.NotFound(w, r)
        return
    }

    // Redirect to the original long URL
    http.Redirect(w, r, urlObj.LongUrl, http.StatusSeeOther)
}
```

Purpose: Retrieves the original URL associated with the provided shortcode and redirects to it

6. CreateShortenUrl() Function

```go
func CreateShortenUrl(w http.ResponseWriter, r *http.Request) {
    if r.Method != http.MethodPost {
        http.Redirect(w, r, "/", http.StatusSeeOther)
        return
    }

    if err := r.ParseForm(); err != nil {
        http.Error(w, "Invalid input", http.StatusBadRequest)
        return
    }
    longUrl := r.FormValue("url")

    // Generate a shortcode
    shortCode := utils.GenerateShortCode(6) // Adjust the length as needed

    // Create the UrlObject
    urlObj := model.UrlObject{
        LongUrl:   longUrl,
        ShortCode: shortCode,
    }

    // Insert into MongoDB
    insertShortenUrlRecord(urlObj)

    // Redirect back to the homepage with the shortened URL
    tmpl := template.Must(template.ParseFiles("templates/index.html"))
    tmpl.Execute(w, map[string]interface{}{
        "ShortenedURL": fmt.Sprintf("http://localhost:8080/%s", shortCode),
    })
}
```

Purpose: Retrieves the original URL associated with the provided shortcode and redirects to it