

# Deliverable report for GolfApp

---

**Henry Fry**

**Student No: 15387755**

**Student account: 5fryh55**

**GolfApp can be found live here : [GolfApp](#)**

**GolfApp GitHub repository can be found live here : [GitHub](#)**

## Introduction

The original idea behind GolfApp was to provide a way to keep track of all the scores and rounds of golf myself and my friends play. Currently, we score all our cards on paper, add them up and then dispose of them, or use the notes app on someone's phone to store the scores. This isn't readily accessible to all, doesn't give us an easy way to look back on scores or track improvements. So, the purpose of GolfApp is to provide a way to store our historical and future scorecard data. This can even be expanded on in the future to include Handicap calculations, something that is currently something that must be paid for by signing up for a club.

## System Overview

The program uses a mongoDB database to store collections of data. These collection are scorecards, the raw data, golfers, courses and users. The scorecard collection stores the golfer, the course, a score for each hole and a total. The golfer and courses collections are simple in nature, as they are only storing one or two values. These will be used to create scorecards, but they are also editable.

A scorecard requires an existing golfer and an existing course as the create scorecard form is populated with a view that pulls golfers and course names. The scorecard model shows the makeup of objects in the collection, also how a scorecard is attached to a golfer and course id which will be useful for later use.

```
const scorecardSchema = new Schema({
  name:{ type: String, required: [true, "Name is required"]},
  course:{ type: String, required: [true, "Course is required"]},
  date:{ type: Date, required: [true, "Date is required"]},
  hole_1: { type: Number, required: [true, "Score is required"]},
  hole_2: { type: Number, required: [true, "Score is required"]},
  hole_3: { type: Number, required: [true, "Score is required"]},
  hole_4: { type: Number, required: [true, "Score is required"]},
  hole_5: { type: Number, required: [true, "Score is required"]},
  hole_6: { type: Number, required: [true, "Score is required"]},
  hole_7: { type: Number, required: [true, "Score is required"]},
  hole_8: { type: Number, required: [true, "Score is required"]},
  hole_9: { type: Number, required: [true, "Score is required"]},
  hole_10: { type: Number, required: [true, "Score is required"]},
  hole_11: { type: Number, required: [true, "Score is required"]},
  hole_12: { type: Number, required: [true, "Score is required"]},
  hole_13: { type: Number, required: [true, "Score is required"]},
  hole_14: { type: Number, required: [true, "Score is required"]},
  hole_15: { type: Number, required: [true, "Score is required"]},
  hole_16: { type: Number, required: [true, "Score is required"]},
  hole_17: { type: Number, required: [true, "Score is required"]},
  hole_18: { type: Number, required: [true, "Score is required"]},
  score: { type: Number, required: [true, "Total is required"]},
  golfer_id:{
    type: mongoose.Schema.Types.ObjectId,
    ref: "Golfer"
  },
  course_id:{
```


```
        type: mongoose.Schema.Types.ObjectId,  
        ref: "Course"  
      },  
    },  
  
    {timestamps: true}  
  );
```

When the user opens the GolfApp, they are presented the index page. This landing page offers the sign up and sign in buttons to users. Sign in presents a username and password form, the password is hashed using bcrypt and the user is stored in the user collection. The sign in page presents the same form, but will check for existing user and password to log the user in.

Once logged in the full menu appears. All pages are auth protected, meaning that any attempt to reach them without being logged in will redirect the user to the index page. The golfers and courses view pages are similar, they both read data from their respective collections and present this data in a table to the user. Golfers and courses can be edited and deleted in this table. The scorecards view presents the scorecard data, with no edit function. This is a deliberate decision as, in golf once the scorecard is completed and 'signed' it is final.

Each of these 3 pages has a create button. Create golfer requires name, create course requires name and course par. Once created the user is redirected back. Create scorecard presents a larger form, the golfer and course is pulled from a view in scorecardController. This only lets existing golfers and courses to be used. Each hole requires a number, with a min of 1 and max of 15. Each of the fields of the form are "required" this stops users entering blank values into the database.

A basic outline of the site structure:

 System Overview

## Key Design Decisions

As the purpose of the web app is to track scorecards made by myself and friends, I will not be using a pre existing data set. I have created a seeder, that populates the database with entries during development, but these will ultimately be removed before the web app sees use.

### Database entities

Users:

- id, Object id.
- Username, String.
- Password, hashed string.

User is read by the program to access different pages. Sign up writes to the database.

Scorecard:

- id, Object id.
- name, string.

- course, string.
- hole (1-18), number.
- score, number

Scorecard data is read to display, create scorecard writes to database.

Courses:

- id, Object id.
- course, String.
- par, number.

Courses is read by the program to view data and populate create scorecard form. Add course writes to the database.

Golfers:

- id, Object id.
- name, String.

Golfer is read by the program to view data and populate create scorecard form. Add golfer writes to the database.

Golfers and courses are examples of one to many relationships. Each can have many scorecards attached to them, while a course can have many golfers playing there and a golfer could have played many courses. These relationships could be further explored in future updates to this program to filter scorecards by course or by golfer.

## Security and scalability

The pages on the web app are all auth protected, only being able to be accessed by logged in users. Currently every user has access to the full CRUD capabilities that are offered to users. As the purpose of this app is to be used by friends for collating scorecards and courses, each should be able to manipulate data this way. However, with a larger user base actions should be restricted to "admins" to reduce malicious data manipulation. Manually populating the database with approved users and removing sign up functionality would solve this.

The models for courses, golfers, scorecards and users all make use of validity and required parameters. For instance, in the form to create a scorecard, the input in each field is required, and scores are limited to a certain range. This makes sure all scorecard data is consistent. The same goes for adding golfers and courses, each requires an input and blank values are rejected.

The scorecard model attaches the course id and the golfer id to the entry. This will be useful in future scaling up, for instance potentially adding filters to scorecards so a user can view cards based on course or golfer.

## Conclusion

The main aim of the project, to create a web app that can be used to store golf scores with my friends, has been achieved. However, the process brought up a number of new ideas and challenges that weren't able to be included. For instance, dynamically updating total scorecards for each golfer as new scorecards are created. I found that as the site took shape, these ideas became harder to implement in the time frame I had to deliver

the complete project. So, while I was not able to implement this feature, I did incorporate the groundwork necessary to develop them in the future. The potential to take the scores for each user and bring in Tabling libraries such as D3 to track individuals progress also presents an interesting future challenge to further the functionality of the web app.