H446/03

Callum Tarttelin, 1164, 51617

Table of Contents

1. Research

The client currently has a bowling league with an older website. The purpose of this project will be to replace this website with a website easier to use.

1.1. Emails

I received an email from the client about what the project will need to do.

1.1.1. The Email

Welcome to the seemingly easy world of tenpin bowling league management
All of this is currently managed through a number of Excel worksheets linked together, with a load of macros to produce the static web pages and the scoresheets for each week with the updated averages, handicaps and league standings. The web pages are then uploaded to the website each week. It would be great to get to a point where the data entry can be done directly into a [database] on the web server so it can be done from anywhere (and in theory, by anyone). This would help when I am on holiday. Then we can look at webpages that don't need to be uploaded, they would simply get data out of the database. I think it would make sense to meet up so I can show you the mess I work with (which causes headaches at the start of each season when I tweak it to fit – and inevitably break something). That way you'll also be able to ask "why on earth do you do that", or "you never mentioned that". Maybe I can show you how I setup a new season, which is when I really find out the bits I need!

1.1.2. Background

The basic structure is that we have a number of teams competing over a number of weeks which make up a season. Each team can have up to 9 bowlers registered and actively bowling. A bowler can switch from one team to another mid-season, however they can only move once during the season. When this happens, their average and handicap move with them. This happens rarely but does happen.

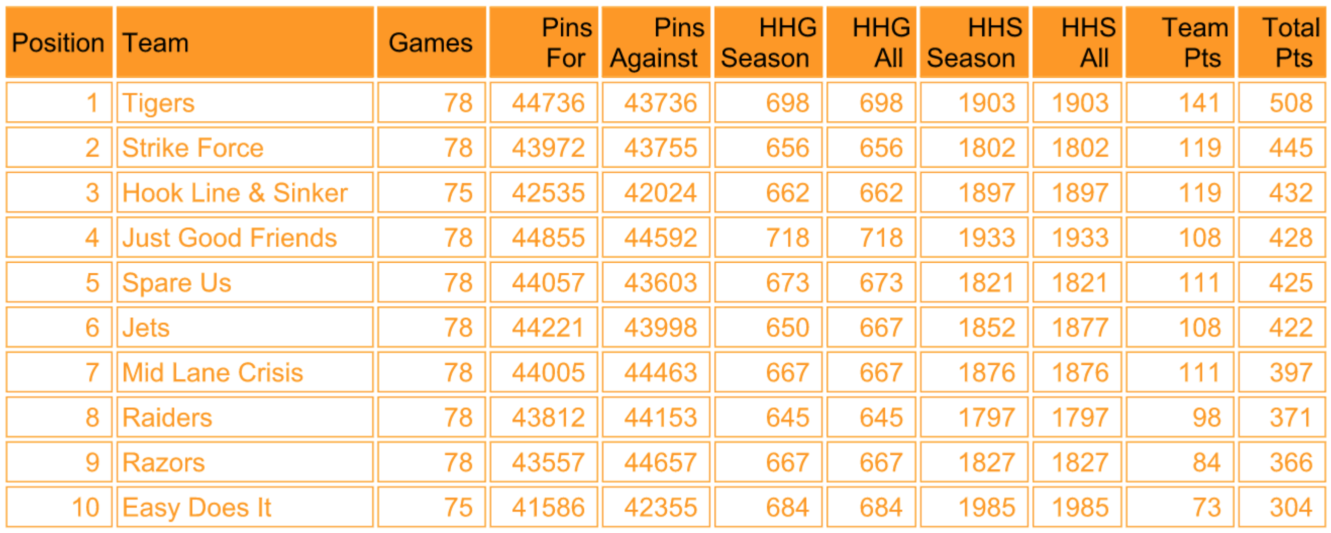
1.1.3. Handicapping

The league runs a handicapping system providing additional points to a bowler based on their current average score. The calculation that we use in Excel is =INT(200-INT(bowler_average*0.8) There is a maximum handicap of 80 and a minimum of 0. A bowler with an average of 180 has a handicap of 16 ((200-180)*0.8), an average of 180.9 is the same. A bowler with an average of 90 has a handicap of 80 ((200-90)*0.8) = 88 (more than the maximum) A new bowler joining the league will not have an average, hence no handicap; they will receive a handicap based on their first night's scores. So, if Lucy, in the example below, had bowled those scores on the first night, her handicap would have been 64 based on her average of 120. This then is applied to her scratch scores – she would have won her individual match 6-2 instead of losing 3-5! To keep the handicap current, we use the last 24 scratch game scores to calculate the average for each bowler. Just to be confusing, we do also note the average for all league matches through the season (this is used to determine the "high season average" award – but enough of awards! The "blind" score is simply calculated by adding the bowler's average to their handicap and rounding down… average of 120 = handicap of

64 = blind score of 164 which they would need to beat to win points if the opponent didn’t turn up.

1.1.4. Scoring

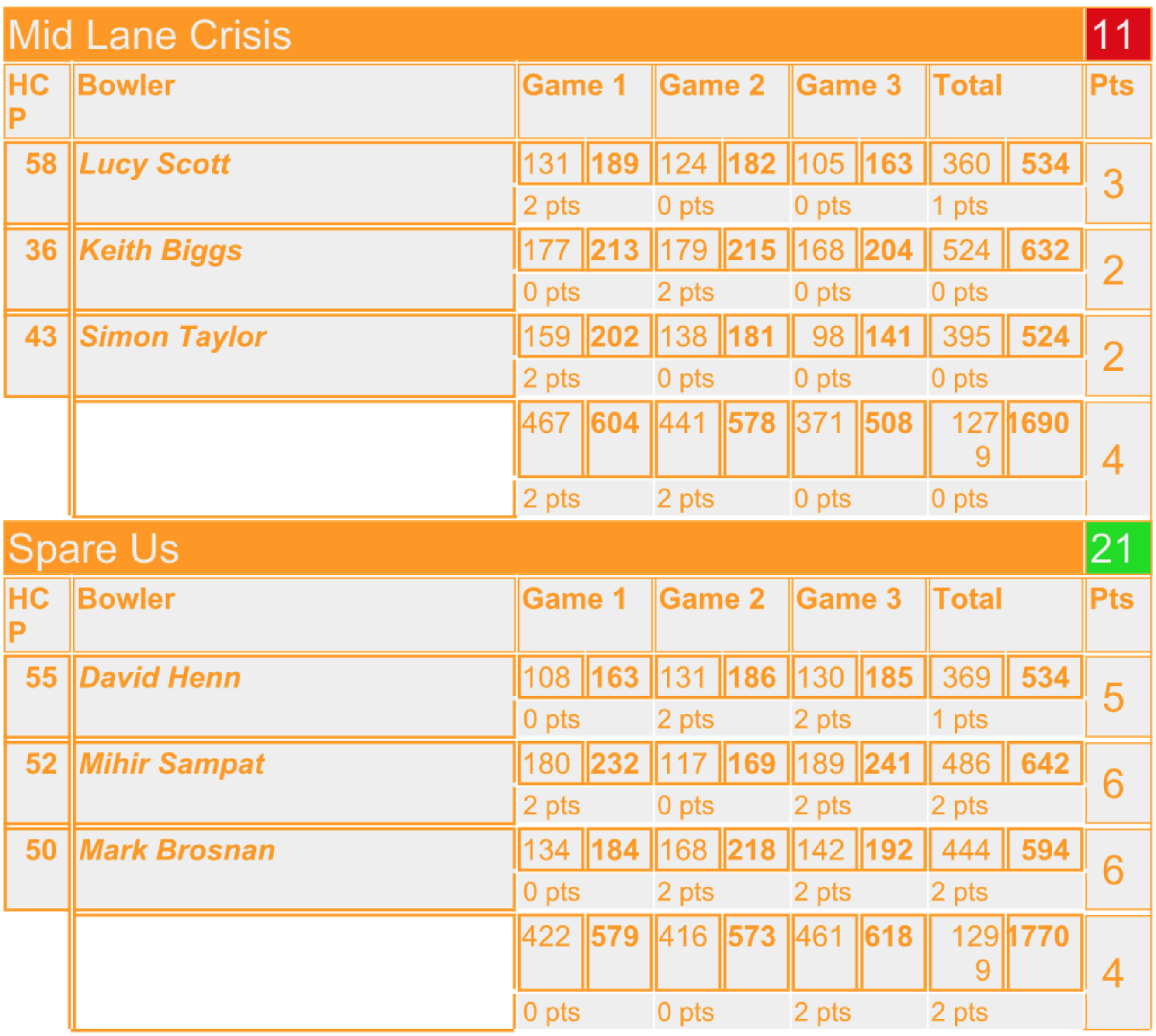
Each week 3 bowlers per team take part and play 3 games each. In reality, it is more complicated or flexible than this. It could be that each game is played by a different bowler, in the event people get injured; so we need the flexibility for each game to be attributed to a different bowler with a different handicap. Example scoresheet (from the website)



Position	Team	Games	Pins For	Pins Against	HHG Season	HHG All	HHS Season	HHS All	Team Pts	Total Pts
1	Tigers	78	44736	43736	698	698	1903	1903	141	508
2	Strike Force	78	43972	43755	656	656	1802	1802	119	445
3	Hook Line & Sinker	75	42535	42024	662	662	1897	1897	119	432
4	Just Good Friends	78	44855	44592	718	718	1933	1933	108	428
5	Spare Us	78	44057	43603	673	673	1821	1821	111	425
6	Jets	78	44221	43998	650	667	1852	1877	108	422
7	Mid Lane Crisis	78	44005	44463	667	667	1876	1876	111	397
8	Raiders	78	43812	44153	645	645	1797	1797	98	371
9	Razors	78	43557	44657	667	667	1827	1827	84	366
10	Easy Does It	75	41586	42355	684	684	1985	1985	73	304

In the match above Lucy Scott is competing directly against David Henn (first bowler for each team). Their individual match is made up of 4 elements; game 1, game 2, game 3 and the series. I currently record the scratch score (131 for Lucy and 108 for David), their handicap and the bowler name. Once the handicap is added to the scratch score we get the handicapped score, this is what is compared to determine who won the points. In this case Lucy’s handicapped score is 189 which beats David’s handicapped score of 163, so Lucy is awarded 2 points for game 1. The points are awarded for games 2 and 3 and finally for the series totals. In this instance because the scores are tied (534) they both are awarded 1 point. In theory, we could calculate the handicap “on the fly”, however recording the handicap as a static value it allows for the anomalies we see at various times during a season (postponement of matches). Score is each 2 points for a win,1 point for a draw (equal score). 32 points are available each week: Each bowler 2 points per game (3 games) plus 2 points for the total (series) score with handicap. Scored against their equivalent bowler on the opposite team (based on who bowls 1st/2nd/3rd). So each bowler can score up to 8 points The team total counts as another pseudo-bowler and is scored the same way – total per game and grand-total The handicap score (meaning the bowlers actual or “scratch” score plus their handicap) is always used to work out who won/lost/tied.

1.1.5. League Standings



Mid Lane Crisis										11
HCP	Bowler	Game 1		Game 2		Game 3		Total		Pts
58	*Lucy Scott*	131	**189**	124	**182**	105	**163**	360	**534**	3
		2 pts		0 pts		0 pts		1 pts		
36	*Keith Biggs*	177	**213**	179	**215**	168	**204**	524	**632**	2
		0 pts		2 pts		0 pts		0 pts		
43	*Simon Taylor*	159	**202**	138	**181**	98	**141**	395	**524**	2
		2 pts		0 pts		0 pts		0 pts		
		467	**604**	441	**578**	371	**508**	1279	**1690**	4
		2 pts		2 pts		0 pts		0 pts		

Spare Us										21
HCP	Bowler	Game 1		Game 2		Game 3		Total		Pts
55	*David Henn*	108	**163**	131	**186**	130	**185**	369	**534**	5
		0 pts		2 pts		2 pts		1 pts		
52	*Mihir Sampat*	180	**232**	117	**169**	189	**241**	486	**642**	6
		2 pts		0 pts		2 pts		2 pts		
50	*Mark Brosnan*	134	**184**	168	**218**	142	**192**	444	**594**	6
		0 pts		2 pts		2 pts		2 pts		
		422	**579**	416	**573**	461	**618**	1299	**1770**	4
		0 pts		0 pts		2 pts		2 pts		

The league table page shows

- **HHG*** = High Handicap Game (the total team score not per bowler)
- **HHS*** = High Handicap Series (series = sum of the 3 games played any week, again for the team total)
- **Pins for** = total of the scratch scores scored by the team to date
- **Pins against** = total of their oppositions scores each week to date
- **Team Pts** = total points won by the team pseudo bowler each week
- **Total Pts** = all points won by the team
- *There is a "Season" and "All" value for each of these – based on the fact we run a cup competition each season.

That covers the basics of the scoring and how points are allocated. From all of this, I provide a large number of stats (because I am go through phases of being interested in it.

The Cup

The cup competition that runs for a few weeks during the year is also managed this way – but more as an afterthought. The mechanism for determining points can be different to the normal league depending upon how we manage it; normally it's different when we have an odd number of teams. For each team we see a summary of the score each week and some individual achievements. There is a great deal of detail (which I have put together because it was available)… we can make this "phase 2" Ian's comments I haven't addressed elsewhere

Some weeks teams will agree to postpone their game. They catch up the missed games later in the season

- This is where the static handicap is useful, with the option to override it. Bowler's stats are rolled over between seasons (so they continue with the average from last season at the start of a new season)
- Their last 24 games are used to roll their average to the new season. If they haven't bowled 24 games, then up to 24 games are carried over; if it's less than 6 games then they will start as if new. All achievements are carried over (awards for a 200 game, or 6 consecutive strikes)
  There could be a different number of teams some seasons
- If it's an odd number the league make the decision to either bowl against the "blind", or to have a week of no bowling. A team could drop out mid-season and scores must be removed from each weeks results (is that right Phil?)
- It depends, I will check the constitution; if they have bowled against each team just once but a couple of teams twice, we might remove the extra couple of results. We normally encourage them to bowl to a natural break point, but it's not always possible. At the start of the season the system needs to generate a list of matches – which team is playing which other team and on which lanes. The teams need to be put on each lane an equal number of times as far as possible and play each other team twice (maybe 3 times if the number of teams is low enough and there are enough weeks available). If the league has 10 teams assume lanes 1 to 10 are used each week
- We are typically a 10 team league and to be awkward we use lanes 3 to 12 but if we have 10 lane identifiers, we can modify them to be the lanes we actually use.
  Some weeks are used for non-league matches (special competitions as teams or individuals tournaments). Some weeks each summer there is no bowling (too maybe people on holiday so we stop). Also bank holidays there is no playing.
- We need a mechanism for recording these scores (we currently use a different area of the spreadsheet to record these, so they are not included within the range used for calculating averages and handicaps but within the range used for awards.

1.2. Next stuff

On top of this existing criteria the current project should have a login system to allow users to add scores for their games. This will need to be confirmed by the other team.

2. Planning

2.1. What objects, and what do they have

```
League{
    teams: [Team, Team ...],
    rota: [Game, Game ...]
    ranking: computed
}

Team{
    name: String
    Image: Image
    players: [Player, Player ...]
    score: Score Object
}

Player{
    name: String
    score: Score Object
}

User{
    ID: 47q047309-47120-97410-298490
    team: Team
    player: Player
    rank: leagueAdmin || teamOwner || scoreAdder
}

Game{
    score: Score Object
    time: yyyy/mm/dd
    venue: Venue
    status: complete || in progress || not started
}
```

2.2. Technologies

I will use springboot and java for the backend. I will use this as java runs well on many platforms. Springboot makes it easy for me to add things into the project. The front end will be written in React JS. I used this as it is very easy to find documentation and sources on how to write it. I will also use material design for styling. The database structure will probably be best in a relational database so a springboot SQL database will be used such as H2 and JPA to communicate nicely. I will use nightwatch for integration testing. This will allow me to automatically use the website, expect behaviour, and take screenshots during. I will use a python script to put all the testing data into the writeup automatically. To allow easy script modification of the writeup it will be written in asciidoc and made into a pdf with asciidoctor-pdf, the source is plain text and can be easily modified. It supports code highlighting etc as well.

2.3. Design

Upon talking to the client it was established that all original features of the site were important.
There will be support for multiple types of tournament including elimination and round robin brackets.
A logged in user will be defaulted to a page of their and their teams statistics.
The style is not important other than the website should be mobile compatible.
There should be support for multiple leagues.
To start a new season of a league it should duplicate the previous and delete parts. To move players only a leagueAdmin can move them.
A shortcut should be made to use the API without a frontend to allow for scripts to do a task faster.

2.4. Testing

Testing will be automated by use of unit testing and nightwatch.
Nightwatch will be used in order to test the entire application by it's user interface and report on wheter or not it is working as intended. It will also take screenshots of the application to allow a developer to quickly look over these screenshots as opposed to having to navigate the website. This allows for quick testing of all versions without the necessity for user input.
Unit testing will be done to check the functionality of functions. If a function works correctly the test will pass and will then be reported as such. This allows for observation of individual functions to find where errors are occuring.
The data will then be written into the writeup by script in order to have a repeated structure of testing.

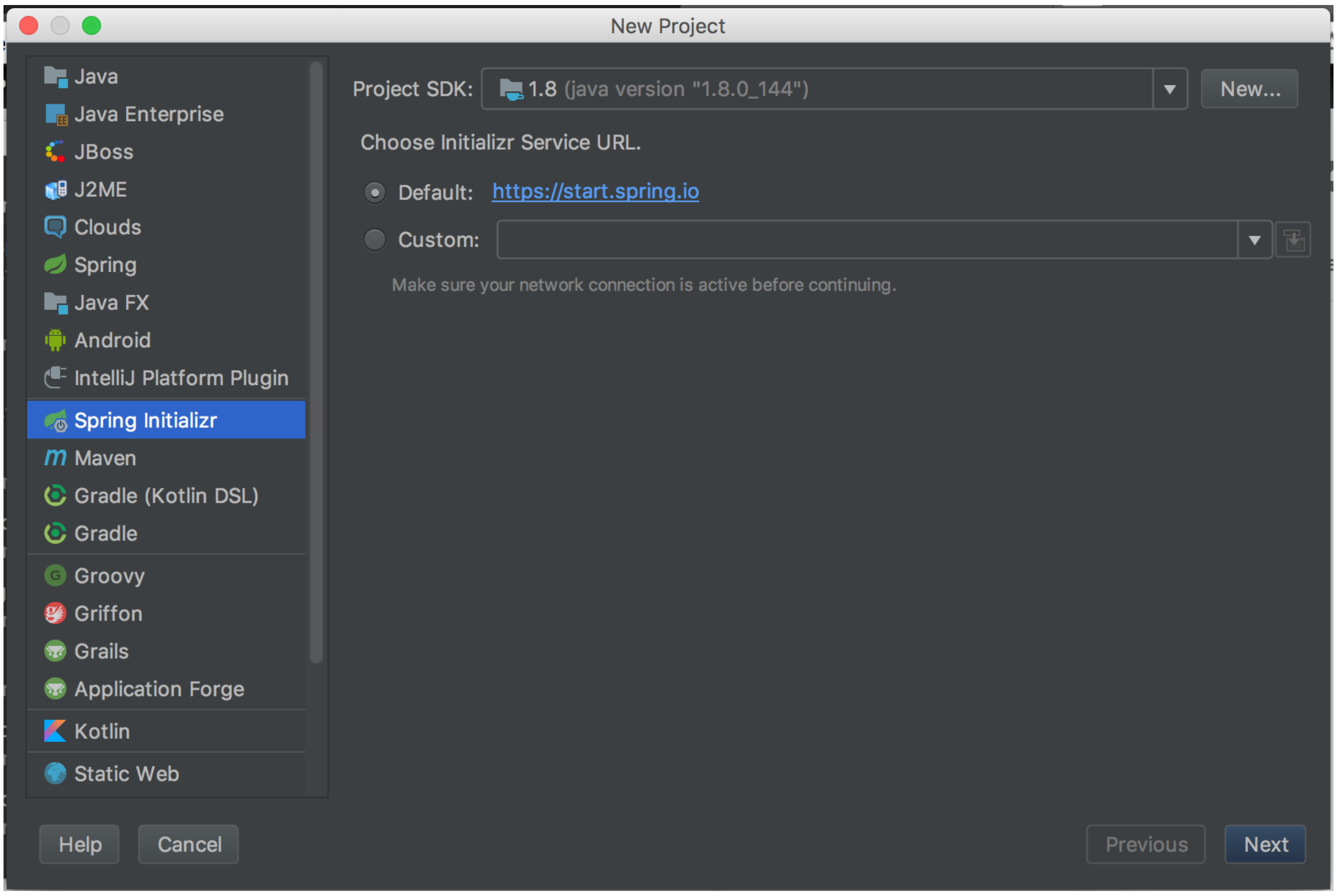
2.5. Algorithms

Adding a new season
Copy the old season as a league, change the start and end times, regenerate the rota, allow user to change the rest, Link to old season data.

3. Development

3.1. Initial Setup

Using IDEA and springboot I can generate a springboot project with certain libraries pre added.



Project SDK: 1.8 (java version "1.8.0_144")

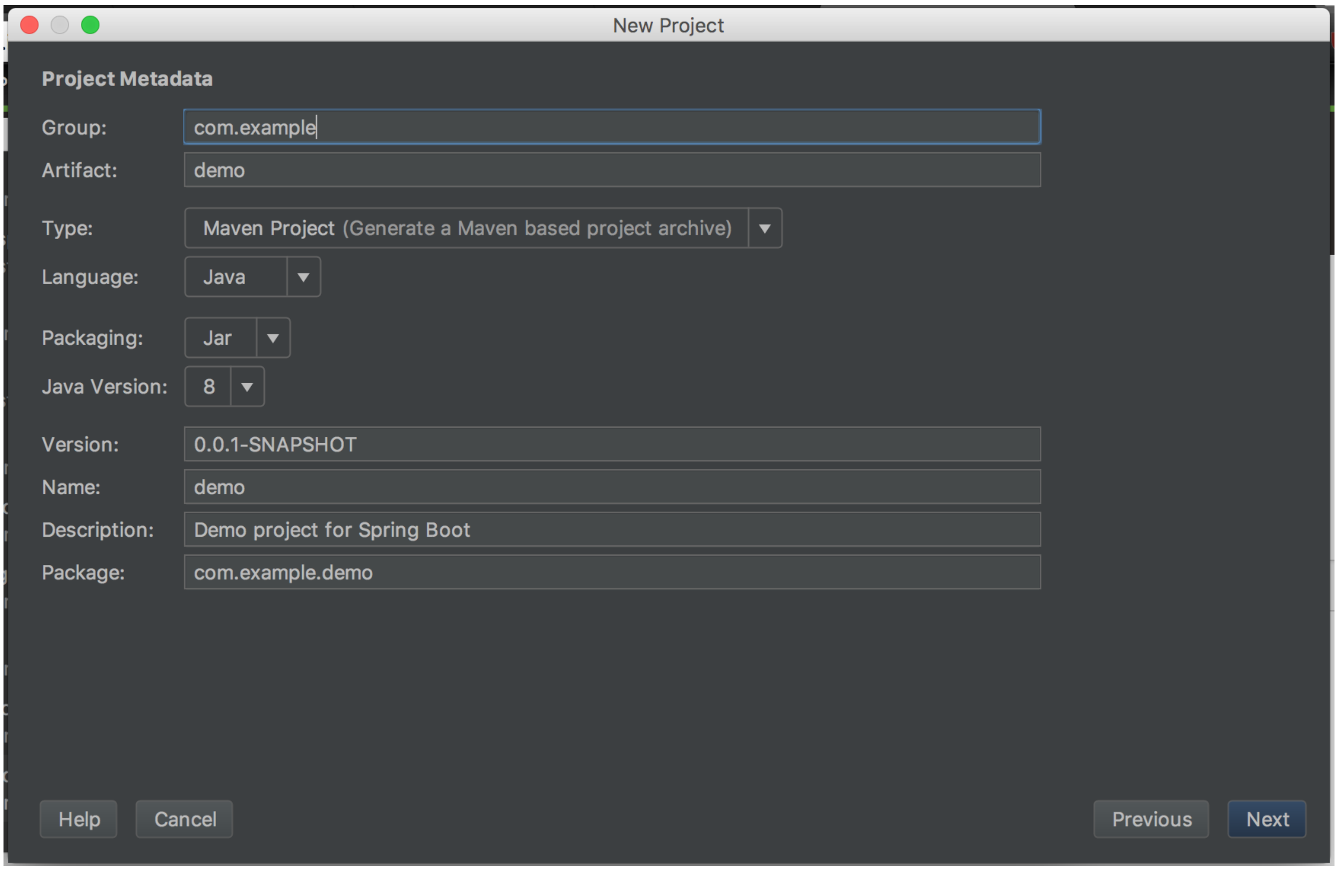
Choose Initializr Service URL.

Default: https://start.spring.io

Custom:

Make sure your network connection is active before continuing.

Step 1: New ProjectS



Project Metadata

Group: com.example

Artifact: demo

Type: Maven Project (Generate a Maven based project archive)

Language: Java

Packaging: Jar

Java Version: 8

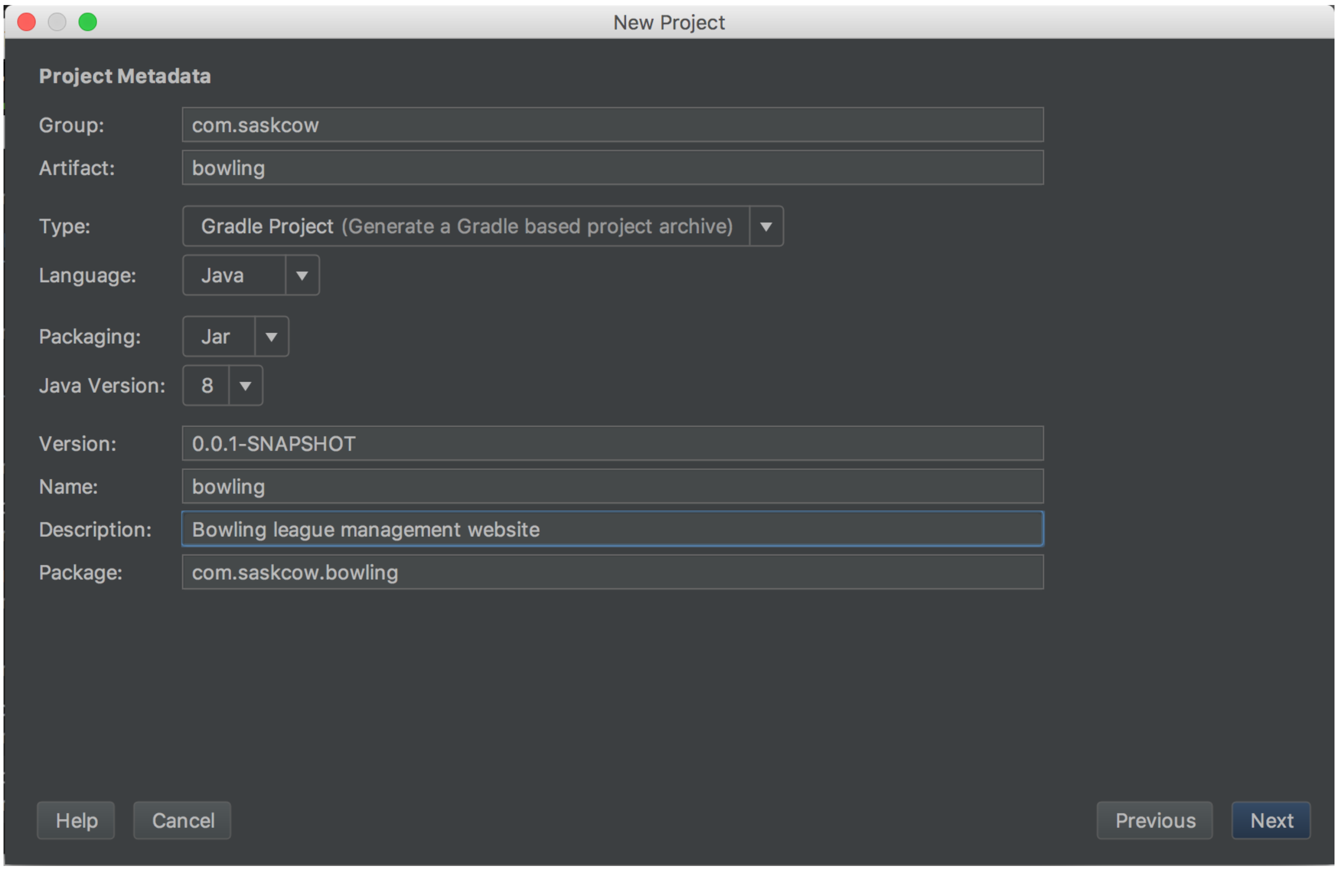
Version: 0.0.1-SNAPSHOT

Name: demo

Description: Demo project for Spring Boot

Package: com.example.demo

Step 2: Springboot Project



Project Metadata

Group: com.saskcow

Artifact: bowling

Type: Gradle Project (Generate a Gradle based project archive)

Language: Java

Packaging: Jar

Java Version: 8

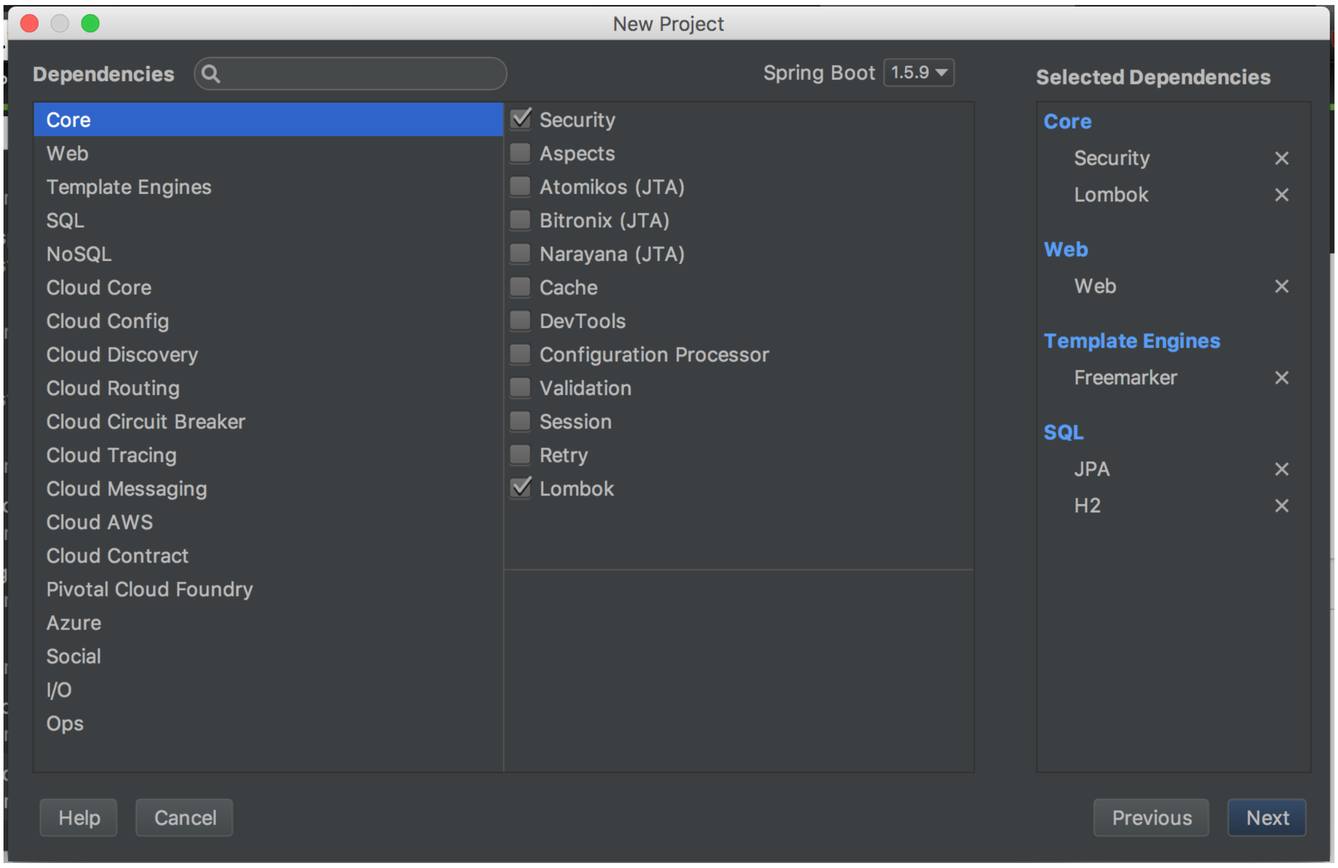
Version: 0.0.1-SNAPSHOT

Name: bowling

Description: Bowling league management website

Package: com.saskcow.bowling

Step 3: Setting up metadata



✓ Security
Aspects
Atomikos (JTA)
Bitronix (JTA)
Narayana (JTA)
Cache
DevTools
Configuration Processor
Validation
Session
Retry
✓ Lombok

Selected Dependencies

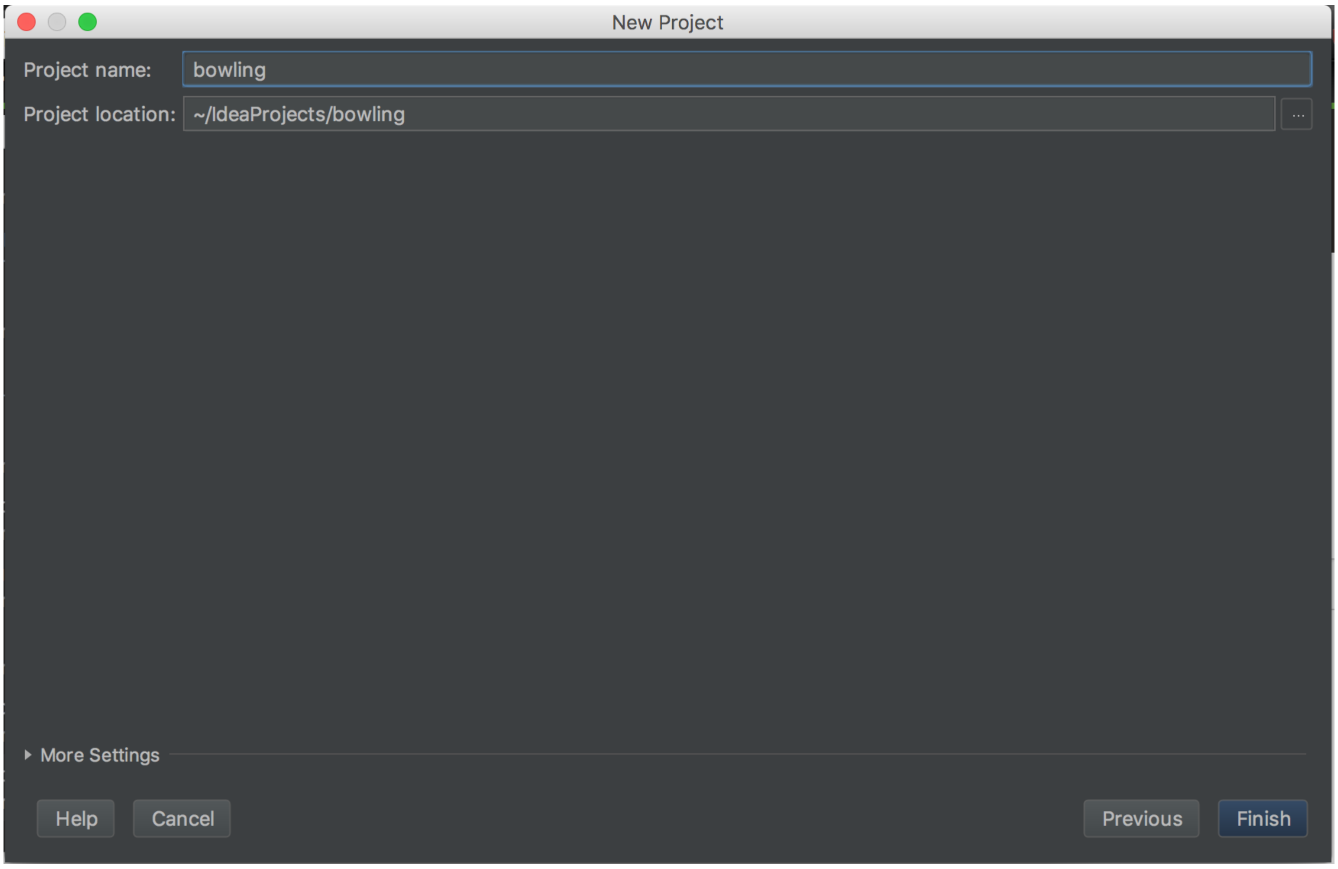
Core
- Security
- Lombok

Web
- Web

Template Engines
- Freemarker

SQL
- JPA
- H2

Step 4: Choosing dependancies



New Project

Project name: bowling

Project location: ~/IdeaProjects/bowling

Step 5: Naming Project

This creates us a simple springboot project with a premade build.gradle and a BowlingApplication and BowlingApplicationTests.

Theses can be left alone for now

3.2. Modelling The Objects

We already decided what objects should do what, so now we add most of the objects To start not all the objects were added so core functionality could be completed first.

Inside the src/main/java/ folder we go into com.saskcow.bowling, this is where the domain package is made inside this is where the objects will be stored, each object should have most of the features in planning Notably scores and users have been omitted in this version.

3.2.1. League.java

Imports condensed

```java
package com.saskcow.bowling.domain;

import ...

@Data
// Creates a getter and setter for each property
@Entity
// JPA annotation, makes it good to store in a database
@NoArgsConstructor
// Creates a constructor with nothing
@AllArgsConstructor
// Creates a constructor with everything
public class League {
    private @Id @GeneratedValue Long id;
    // Generate a long value to be used as ID, always unique
    private String name;
    // Name Property of the class
    @OneToMany(mappedBy = "league", cascade = CascadeType.ALL)
    // This prevents a "Failed to load ApplicationContext" error
    // Additionally the properties of it say if the league is deleted, as are all the
teams
    private List<Team> teams;
    // List<Team> just a list of the teams, type specified in java

    public League(String name, List<Team> teams){
        // A constructor, sets name and teams to what it was created with ID
autogenerated
        this.name = name;
        this.teams = teams;
    }
}
```

⚠ Ensure @ManyToOne etc set to avoid "Failed to load ApplicationContext"

3.2.2. Team.java

Imports condensed

```
package com.saskcow.bowling.domain;

import ...

@Data
@Entity
@NoArgsConstructor
@AllArgsConstructor
//Same as with League
public class Team {
    private @Id @GeneratedValue Long id;
        private String name;
        @OneToMany(mappedBy = "team", cascade = CascadeType.ALL)
        // Players part of team, so a team will change players on its change
        private List<Player> players;
        @ManyToMany
        // Teams have many games, games have 2 teams so a manytomany is identified
        private List<Game> games;
        @ManyToOne
        private League league;

    // A constructor with everything but generating the id
    public Team(String name, List<Player> players, List<Game> games, League league) {
        this.name = name;
        this.players = players;
        this.games = games;
        this.league = league;
    }

    // A second constructor is put in place in order to create a team which has no
players or games
    public Team(String name, League league) {
        this.name = name;
        this.league = league;
        this.players = new LinkedList<>();
        this.games = new LinkedList<>();
    }
}
```

3.2.3. Player.java

Imports condensed
Very similar to previous, nothing new used.

```java
package com.saskcow.bowling.domain;

import ...

@Data
@Entity
@NoArgsConstructor
@AllArgsConstructor
//Same as with League
public class Team {
    private @Id @GeneratedValue Long id;
        private String name;
        @ManyToOne
        private Team team;

        public Player(String name, Team team) {
            this.name = name;
            this.team = team;
        }
}
```

3.2.4. Game.java

Imports condensed
Also very similar to previous, nothing new used.

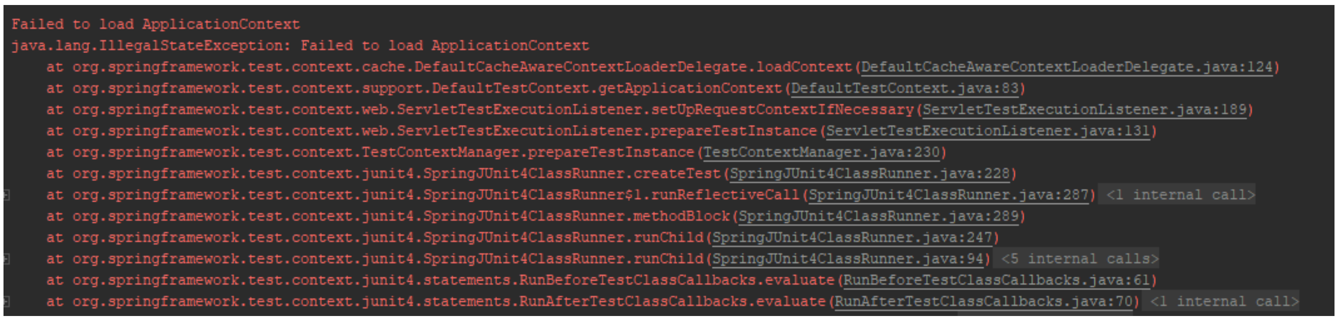
```java
package com.saskcow.bowling.domain;

import ...

@Data
@Entity
@AllArgsConstructor
@NoArgsConstructor
public class Game {
    private @Id @GeneratedValue Long id;
    private LocalDateTime time;
    private String venue;
    @ManyToMany
    private List<Team> teams;
    // Should only ever have 2 values, not enforced

    public Game(LocalDateTime time, String venue, List<Team> teams) {
        this.time = time;
        this.venue = venue;
        this.teams = teams;
    }
}
```

3.2.5. Errors at this stage

This stage was fairly simple so few errors occurred other than occasional mistypes picked up by the ide as it went along. It is also hard to find errors at this stage due to nothing happening.

@ManyToOne, @OneToMany, @ManyToMany annotations

Without these annotations in place a java.lang.IllegalStateException is raised



```
Failed to load ApplicationContext
java.lang.IllegalStateException: Failed to load ApplicationContext
    at org.springframework.test.context.cache.DefaultCacheAwareContextLoaderDelegate.loadContext(DefaultCacheAwareContextLoaderDelegate.java:124)
    at org.springframework.test.context.support.DefaultTestContext.getApplicationContext(DefaultTestContext.java:83)
    at org.springframework.test.context.web.ServletTestExecutionListener.setUpRequestContextIfNecessary(ServletTestExecutionListener.java:189)
    at org.springframework.test.context.web.ServletTestExecutionListener.prepareTestInstance(ServletTestExecutionListener.java:131)
    at org.springframework.test.context.TestContextManager.prepareTestInstance(TestContextManager.java:230)
    at org.springframework.test.context.junit4.SpringJUnit4ClassRunner.createTest(SpringJUnit4ClassRunner.java:228)
    at org.springframework.test.context.junit4.SpringJUnit4ClassRunner$1.runReflectiveCall(SpringJUnit4ClassRunner.java:287) <1 internal call>
    at org.springframework.test.context.junit4.SpringJUnit4ClassRunner.methodBlock(SpringJUnit4ClassRunner.java:289)
    at org.springframework.test.context.junit4.SpringJUnit4ClassRunner.runChild(SpringJUnit4ClassRunner.java:247)
    at org.springframework.test.context.junit4.SpringJUnit4ClassRunner.runChild(SpringJUnit4ClassRunner.java:94) <5 internal calls>
    at org.springframework.test.context.junit4.statements.RunBeforeTestClassCallbacks.evaluate(RunBeforeTestClassCallbacks.java:61)
    at org.springframework.test.context.junit4.statements.RunAfterTestClassCallbacks.evaluate(RunAfterTestClassCallbacks.java:70) <1 internal call>
```

Figure 1. Start of stacktrace

by adding the

```
@ManyToOne
@OneToMany
@ManyToMany
```

this exception is no longer raised allowing it to compile

3.3. Repositories

Repositories in java are very simple to implement, all repositories can be created with *{ObjectName} to be substituted with League, Team, Player and Game*

```
package com.saskcow.bowling.repository;

import ...

// Create a new Repository which copies a CrudRepository so it has all the functions
//<{ObjectName}, Long> shows it stores {ObjectName} by a Long, the Long being the id
of the object.
public interface {ObjectName}Repository extends CrudRepository<{ObjectName}, Long> {
}
```

At this stage this is all that was done for each object and it was saved as {ObjectName}Repository.java inside com.saskcow.bowling.repository

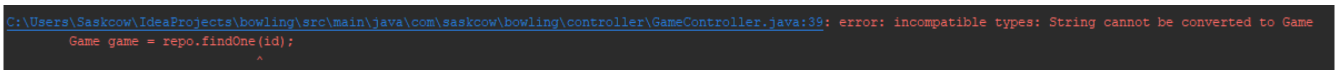
3.3.1. Errors at this stage

Errors this stage were created by trying to figure out whether or not some CrudRepository things

were worth changing
They weren’t.

CrudRepository checking

If the <{ObjectName}, Long> is changed it stores a different type object and returns a different type, so the object cannot be retrieved as itself.



```
C:\Users\Saskcow\IdeaProjects\bowling\src\main\java\com\saskcow\bowling\controller\GameController.java:39: error: incompatible types: String cannot be converted to Game
        Game game = repo.findOne(id);
                                ^
```

Figure 2. Compiler Error

this occurs as a Game object cannot be made from a String, which is the returned object.

3.4. Controllers

There are a lot of controllers in this project, and they are a crucial part to communicate with the frontend to start with, a HomeController will be made to return some basic HTML

3.4.1. index.ftl

This file is where all the front end will be, currently it will just show a blank page, due to spring security, a default password can be set in spring application.properties

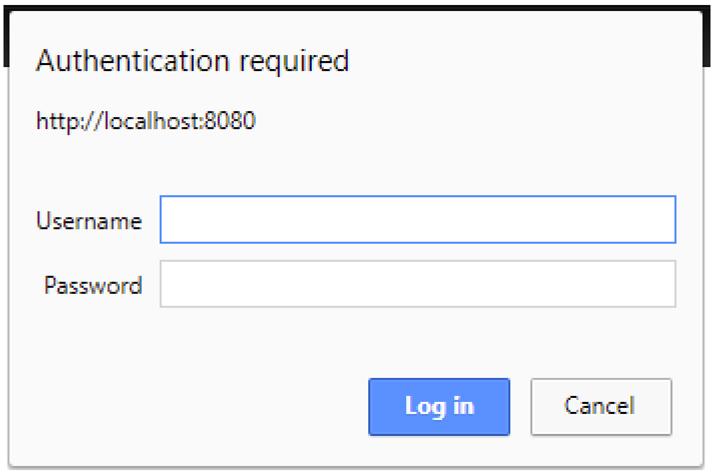


Figure 3. Authentication

```
<html>
<head lang="en">
    <meta charset="UTF-8"/>
    <title>Bowling</title>
    <link rel="stylesheet" href="/style.css"/>
    <link rel="stylesheet" href="/material.min.css"/>
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel
="stylesheet">
    <#--These are in place to get some css to make some elements look slightly nicer--
>
    <#--/style.css is currently blank-->
</head>
<body>

<div id="react"></div>
<#--create a div for ReactDOM to later use-->

<script src="/built/bundle.js"></script>
<#--load the webpack script, this will be created later-->
<#--webpack is what takes all the js and makes it one file-->

</body>
</html>
```

3.4.2. HomeController.java

This will route to the javascript and ftl to sort out the front end

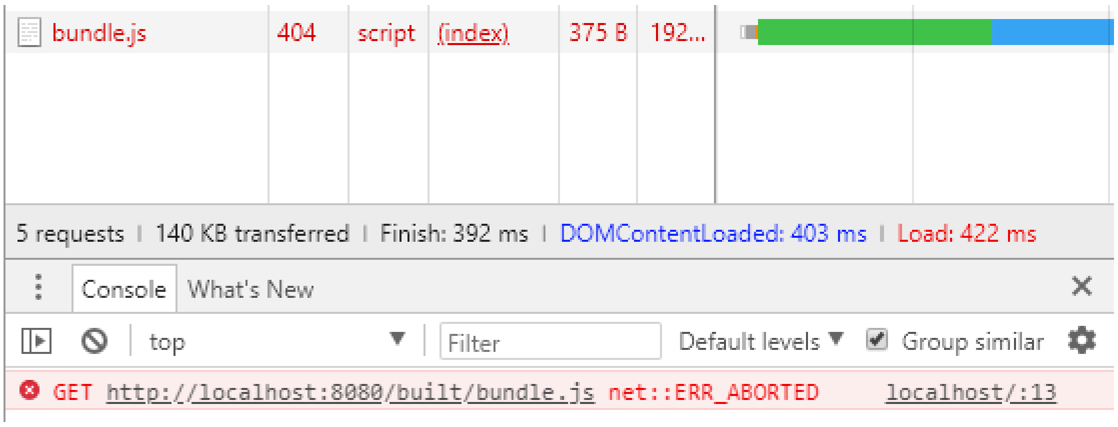
```
package com.saskcow.bowling.controller;

import ...

@Controller
// Look here for request mappings
public class HomeController {

    @RequestMapping(value = {"/"})
    // Any requests to / call this function
    // Later more will be added as more paths are in the front end
    public String index(){
        return "index";
        // Show index file extension guessed, currently .ftl
    }
}
```

3.4.3. Errors at this stage

Only here as they are waiting for future things, which don't exist yet. Or due to user error.

404 Errors

Due to some parts being missing, the browser shows errors in console where it can’t find bundle.js or other parts.

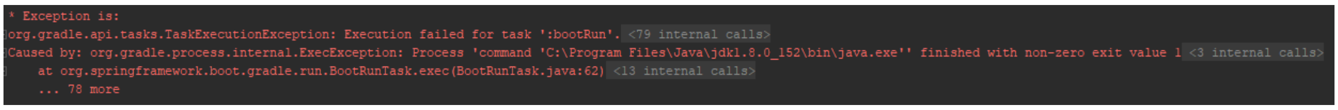


```
bundle.js    404    script    (index)    375 B    192...

5 requests | 140 KB transferred | Finish: 392 ms | DOMContentLoaded: 403 ms | Load: 422 ms

GET http://localhost:8080/built/bundle.js net::ERR_ABORTED      localhost/:13
```

Figure 4. 404 on bundle.js

Multiple Launched Errors

When launched, anything else trying to use the port is blocked, this gives the not especially clear Execution failed for task bootRun as it fails to launch



```
* Exception is:
org.gradle.api.tasks.TaskExecutionException: Execution failed for task ':bootRun'. <79 internal calls>
Caused by: org.gradle.process.internal.ExecException: Process 'command 'C:\Program Files\Java\jdk1.8.0_152\bin\java.exe'' finished with non-zero exit value 1 <3 internal calls>
    at org.springframework.boot.gradle.run.BootRunTask.exec(BootRunTask.java:62) <13 internal calls>
    ... 78 more
```

Figure 5. Launch Interrupted

3.5. League Controller

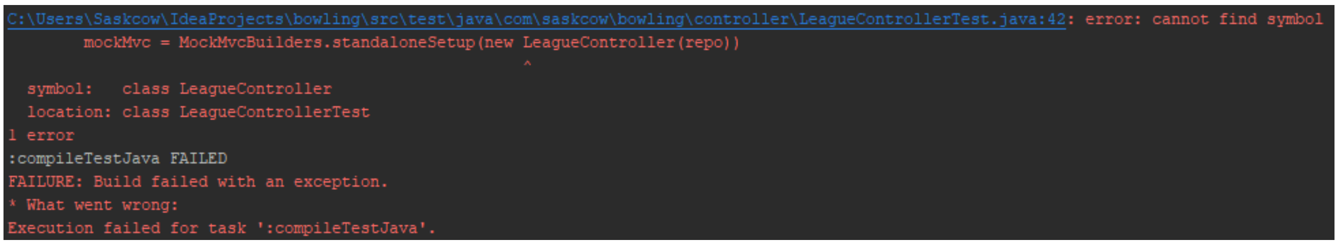
Create mappings for the creating, getting and removing leagues, these will be called from the frontend

3.5.1. LeagueControllerTest.java

Inside the src/test/java I create a folder called controller, this is where I will put controller tests. It is important to test controllers as REST Apis should have consistent features.

```
package com.saskcow.bowling.controller;

import ...

@RunWith(MockitoJUnitRunner.class)
// This runs the tests with a testRunner, this allows assertions which would otherwise
be not allowed
// See Errors below for an example
public class LeagueControllerTest {

    @Mock
    private LeagueRepository repo;
    // Create a LeagueRepository like thing, which does nothing, just pretends it
exists, that's what @Mock does
    private MockMvc mockMvc;
    // Creates a MockMvc to test the api endpoints

    @Before
    //Run before tests
    public void setUp(){
        mockMvc = MockMvcBuilders.standaloneSetup(new LeagueController(repo)).build();
        // Create a LeagueController and run it
    }

    @Test
    // Run this when running tests, ran by MockitoJUnitRunner
    public void addLeague_shouldSaveTheLeague() throws Exception {
        // Doesn't return anything, throws Exception if any part fails, calls test
addLeague_shouldSaveTheLeague
        League league = new League(1L, "Brian", null);
        // Create an instance of a League, the object from earlier
        when(repo.save(isA(League.class))).thenReturn(league);
        // If someone saves a league, return a League, this is what the repo would do,
but the League here is always the same
        when(repo.findOne(league.getId())).thenReturn(league);
        // If someone tries to find this league by its ID, return it
        when(repo.findAll()).thenReturn(Collections.singletonList(league));
        // If someone tries to find all leagues, return this in a list as the only
League

        String uri = mockMvc.perform(post("/api/league")
                // String uri, save the output as a string
                // mockMvc stuff sends a post request to the endpoint
                .content("{\"name\":\"Brian\"}")
                // Send it with this content
                .contentType("application/json"))
                // This content is JSON
                .andExpect(status().isCreated())
                // Should return a 201 (created), if it isn't throw Exception
                .andExpect(header().string("Location",
"http://localhost:8080/api/league/" + league.getId()))
                // Inside the header the location of where the saved object can be
retrieved should be present
                .andReturn().getResponse().getHeader("Location");
                // Save the location header to uri

        mockMvc.perform(get("/api/league"))
                // Send a get request to the endpoint
                .andExpect(status().isOk())
                // Check status is 200 (OK)
                .andExpect(MockMvcResultMatchers.jsonPath("$", hasSize(1)))
                // Check that the JSON is an array with size 1
                .andExpect(MockMvcResultMatchers.jsonPath("$[0].name", equalTo(
"Brian")));
                // Check the first part of the json has a name of "Brian", like the
league earlier

        mockMvc.perform(get(uri))
                // Send a get request to where the location of the league is
                .andExpect(status().isOk())
                .andExpect(MockMvcResultMatchers.jsonPath("$.name", equalTo("Brian")))
                // Check that it has the name Brian
                .andExpect(MockMvcResultMatchers.jsonPath("name", equalTo("Brian")));
                // Check again, by a slightly different method

    }

    @Test
    public void getLeague_shouldFilter() throws Exception {
        League dave = new League(1L, "Dave", null );
        League david = new League(2L, "David", null );
        League brian = new League(3L, "Brian", null);
        // Create 3 leagues
        when(repo.findAll()).thenReturn(Arrays.asList(dave, david, brian));
        // when it calls findByNameContaining("Dav") then it should return all which
have "Dav" in the name
        // findByNameContaining must be added as it is not in CrudRepository
        when(repo.findByNameContaining("Dav")).thenReturn(Arrays.asList(dave,
david));#
        // Same but with Bri
        when(repo.findByNameContaining("Bri")).thenReturn(Collections.
singletonList(brian));

        // We don't do the post request as one already exists
        mockMvc.perform(get("/api/league?name=Dav"))
                // call the endpoint with a query string with name=Dav
                .andExpect(status().isOk())
                .andExpect(MockMvcResultMatchers.jsonPath("$", hasSize(2)))
                // Expect 2 items in the returned array
                .andExpect(MockMvcResultMatchers.jsonPath("$[0].name", equalTo(
"Dave")))
                .andExpect(MockMvcResultMatchers.jsonPath("$[1].name", equalTo(
"David")));
                // Expect that the list is as expected, we only know the order as it
is set earlier
                // Ordinarily order can not be expected

        mockMvc.perform(get("/api/league?name=Bri"))
                // get all leagues with Bri
                .andExpect(status().isOk())
                .andExpect(MockMvcResultMatchers.jsonPath("$", hasSize(1)))
                .andExpect(MockMvcResultMatchers.jsonPath("$[0].name", equalTo(
"Brian")));
                // Check it has 1 item, which is Brian
    }

    @Test
    public void deleteLeague_shouldDeleteLeague() throws Exception {
        doNothing().when(repo).delete(isA(Long.class));
        // When repo.delete is called with an ID, do nothing, nothing at all
        // When this is missing NullPointerExceptions happen
******************************************

        mockMvc.perform(delete("/api/league/1"))
                // Send a delete request
                .andExpect(status().isNoContent());
                // Expect a 204, No Content is returned
        verify(repo, times(1)).delete(1L);
        // Check it actually called delete
        }
}
```

Note at this point there is no controller, so all this fails, and is therefore an error, this is fixed by creating the controller.



```
C:\Users\Saskcow\IdeaProjects\bowling\src\test\java\com\saskcow\bowling\controller\LeagueControllerTest.java:42: error: cannot find symbol
        mockMvc = MockMvcBuilders.standaloneSetup(new LeagueController(repo))
                                                      ^
  symbol:   class LeagueController
  location: class LeagueControllerTest
1 error
:compileTestJava FAILED
FAILURE: Build failed with an exception.
* What went wrong:
Execution failed for task ':compileTestJava'.
```

Figure 6. No Controller Exists

3.5.2. LeagueController

3.5.3. LeagueRepository

findByName ⇒ findByNameContaining

3.5.4. LeagueView

3.5.5. LeagueViewSummary

3.5.6. Errors at this stage

4. Test Formatter

I wrote a test formatter in python to format the xml into asciidoc for the writeup, can't be doing all this manually now can I?

4.1. Research

4.2. Planning

4.3. Development

4.4. Testing

5. Testing

It is important to do testing throughout the project, screenshots and logs start from an early stage in development

Test Run No. 1

JUnit Tests

com.saskcow.bowling.BowlingApplicationTests

1 tests, 0 failed, 0 errors,

Classname	Name	Time
com.saskcow.bowling.BowlingApplicationTests	contextLoads	0.046
com.saskcow.bowling.BowlingApplicationTests		0.046

com.saskcow.bowling.controller.LeagueControllerTest

3 tests, 0 failed, 0 errors,

Classname	Name	Time
com.saskcow.bowling.controller.LeagueControllerTest	deleteLeague_shouldDeleteLeague	0.22
com.saskcow.bowling.controller.LeagueControllerTest	getLeague_shouldFilter	0.126
com.saskcow.bowling.controller.LeagueControllerTest	addLeague_shouldSaveTheCourse	0.115
com.saskcow.bowling.controller.LeagueControllerTest		0.462

com.saskcow.bowling.controller.TeamControllerTest

2 tests, 0 failed, 0 errors,

Classname	Name	Time
com.saskcow.bowling.controller.TeamControllerTest	deleteLeague_shouldDeleteLeague	0.036
com.saskcow.bowling.controller.TeamControllerTest	addLeague_shouldSaveTheCourse	0.076
com.saskcow.bowling.controller.TeamControllerTest		0.112

com.saskcow.bowling.repository.GameRepositoryTest

1 tests, 0 failed, 0 errors,

Classname	Name	Time
com.saskcow.bowling.repository.GameRepositoryTest	thingsSaved_canBeRetrieved	0.067
com.saskcow.bowling.repository.GameRepositoryTest		0.067

com.saskcow.bowling.repository.LeagueRepositoryTest

3 tests, 0 failed, 0 errors,

Classname	Name	Time
com.saskcow.bowling.repository.LeagueRepositoryTest	thingsSaved_canBeRetrieved	0.011
com.saskcow.bowling.repository.LeagueRepositoryTest	thingsSaved_canBeQueried	0.146
com.saskcow.bowling.repository.LeagueRepositoryTest	thingsSaved_canBeDeleted	0.032
com.saskcow.bowling.repository.LeagueRepositoryTest		0.191

Nightwatch Tests

Test times not related to how long the site takes to use

TestLeague

Test Results

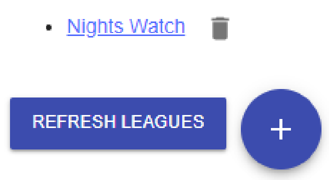
TestLeague

2 tests, 0 failed, 0 errors,

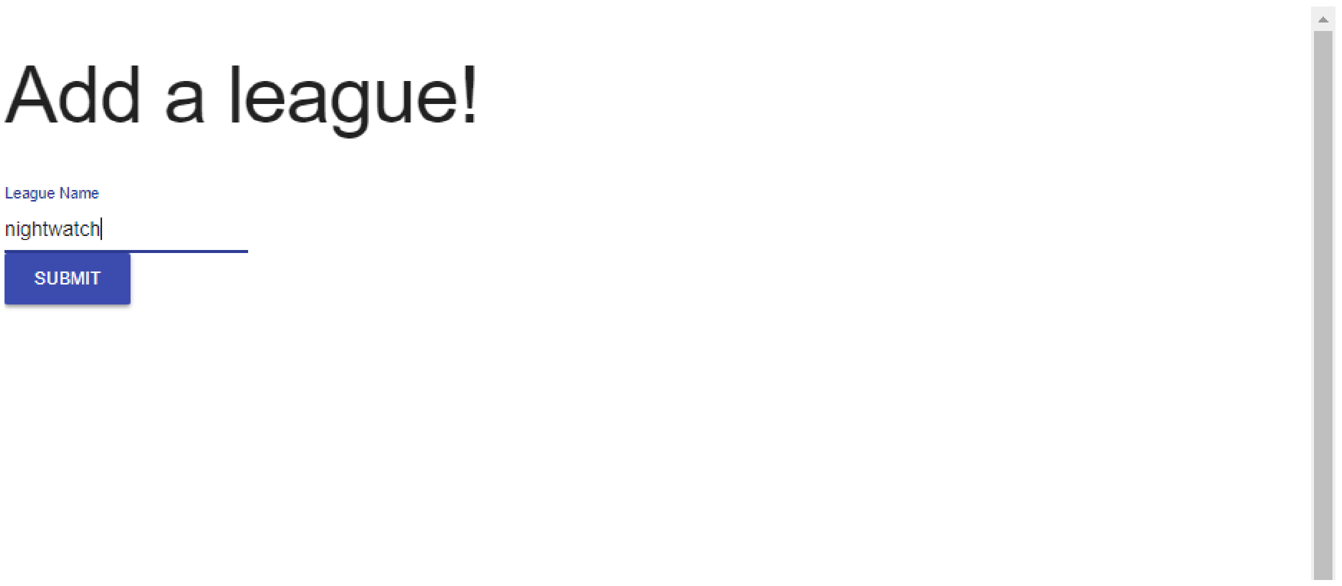
Classname	Name	Time	Assertions
TestLeague	Test adding leagues	7.651	5
TestLeague	Test removing leagues	1.648	3
TestLeague		9.299	

Screenshots

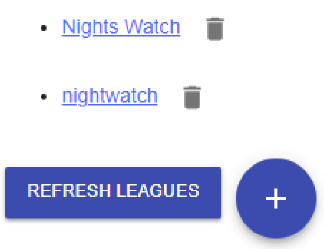
1-start



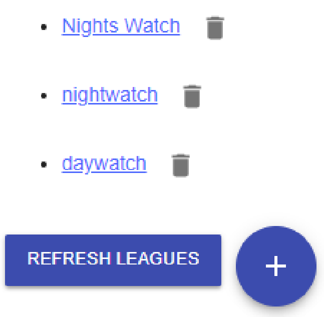
2-adding a league



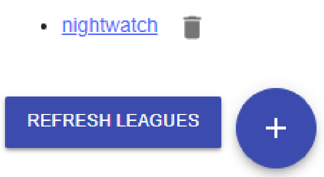
3-Shows League



4-Second League



5-Deleted daywatch



TestTeams

Test Results

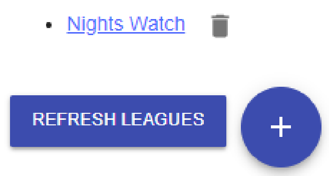
TestTeams

3 tests, 0 failed, 0 errors,

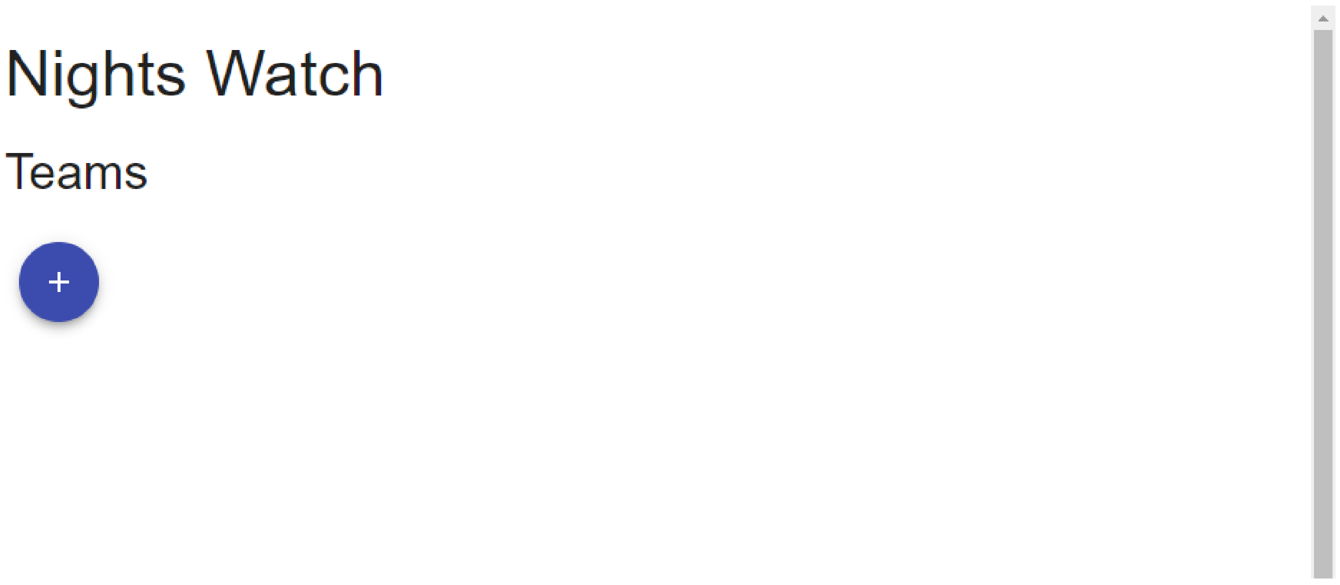
Classname	Name	Time	Assertions
TestTeams	Setup	4.978	3
TestTeams	Test Adding Teams	4.600	6
TestTeams	Deleting	1.786	4
TestTeams		11.36	

Screenshots

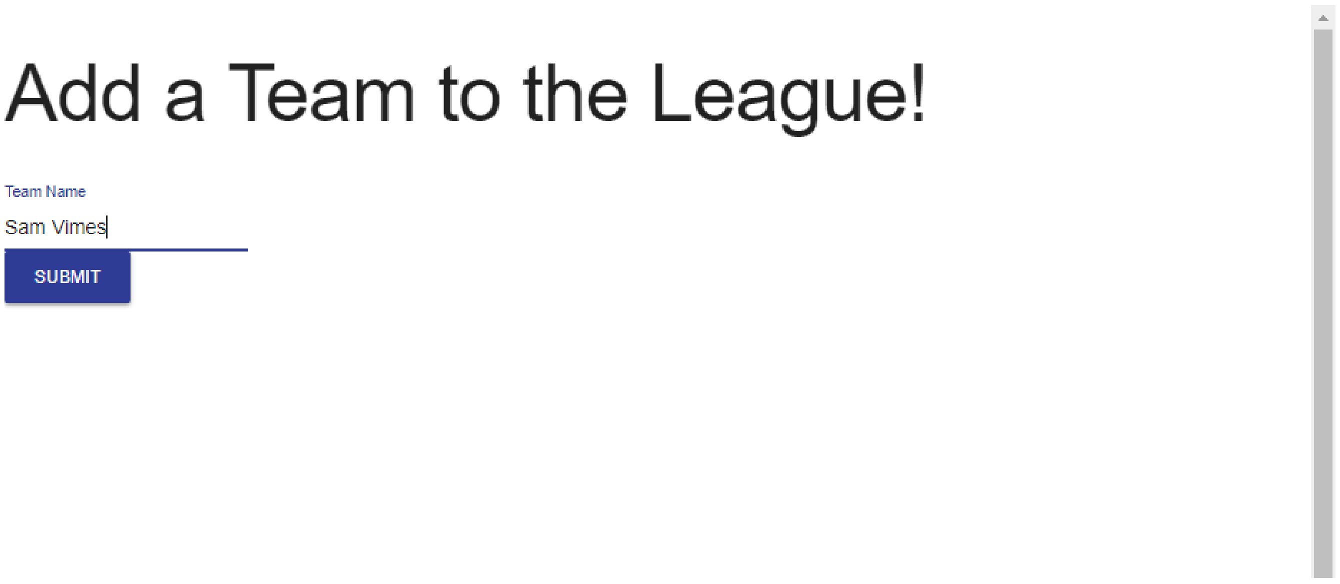
1-init league



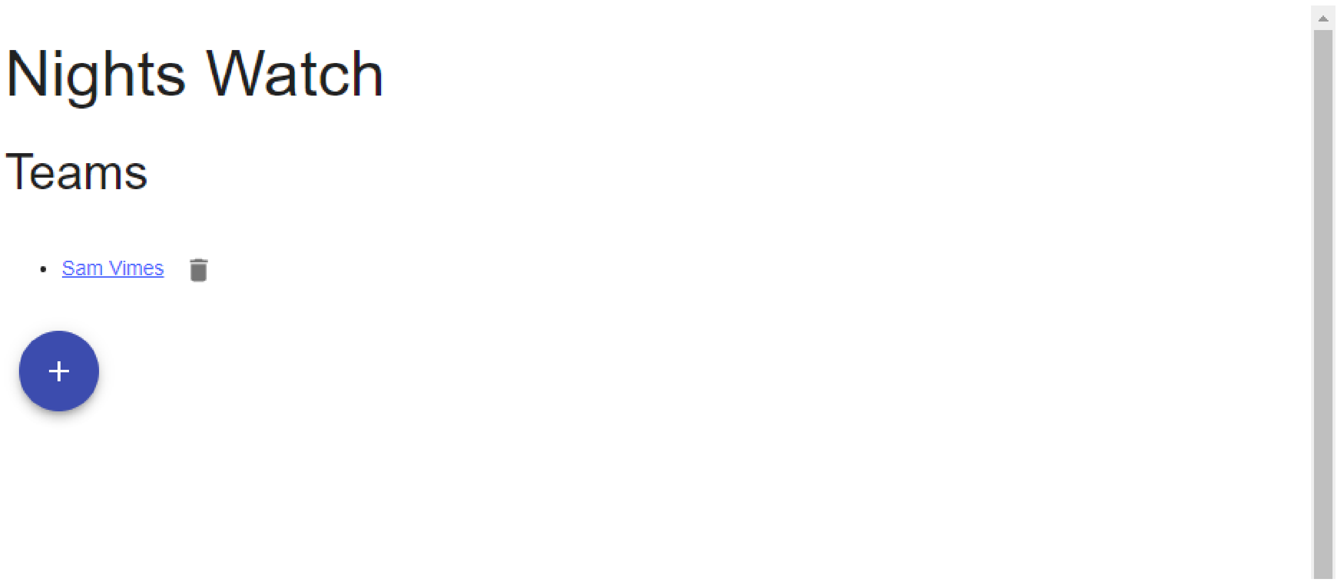
2-League view



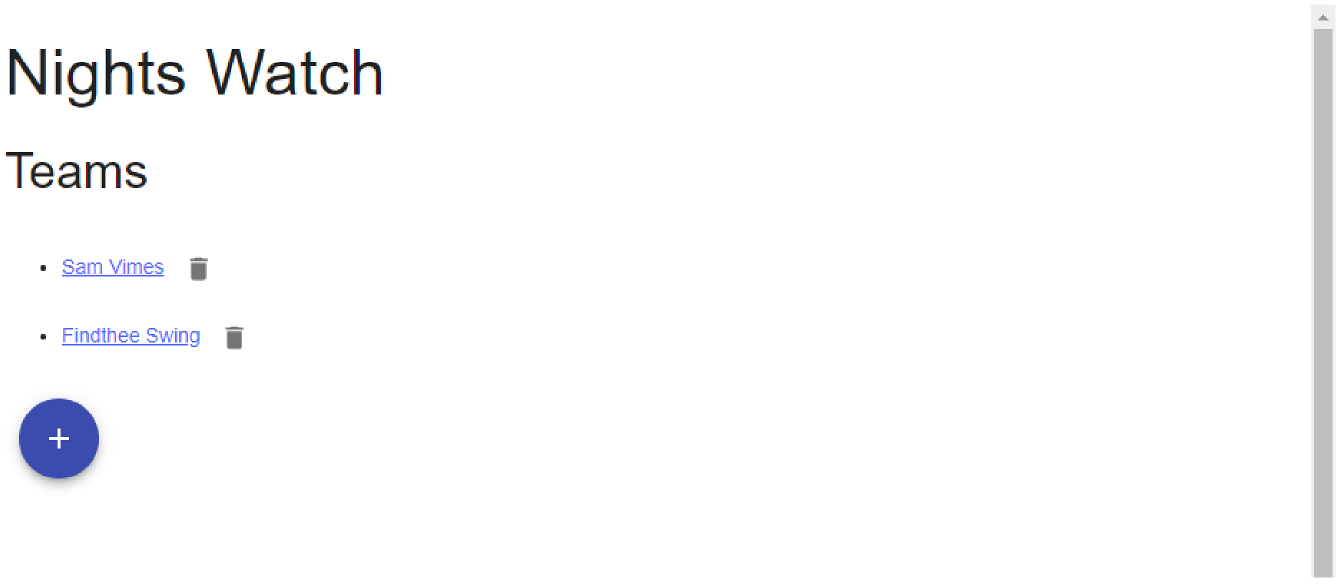
3-Add Team screen



4-Sam Vimes in the watch



5-2 teams

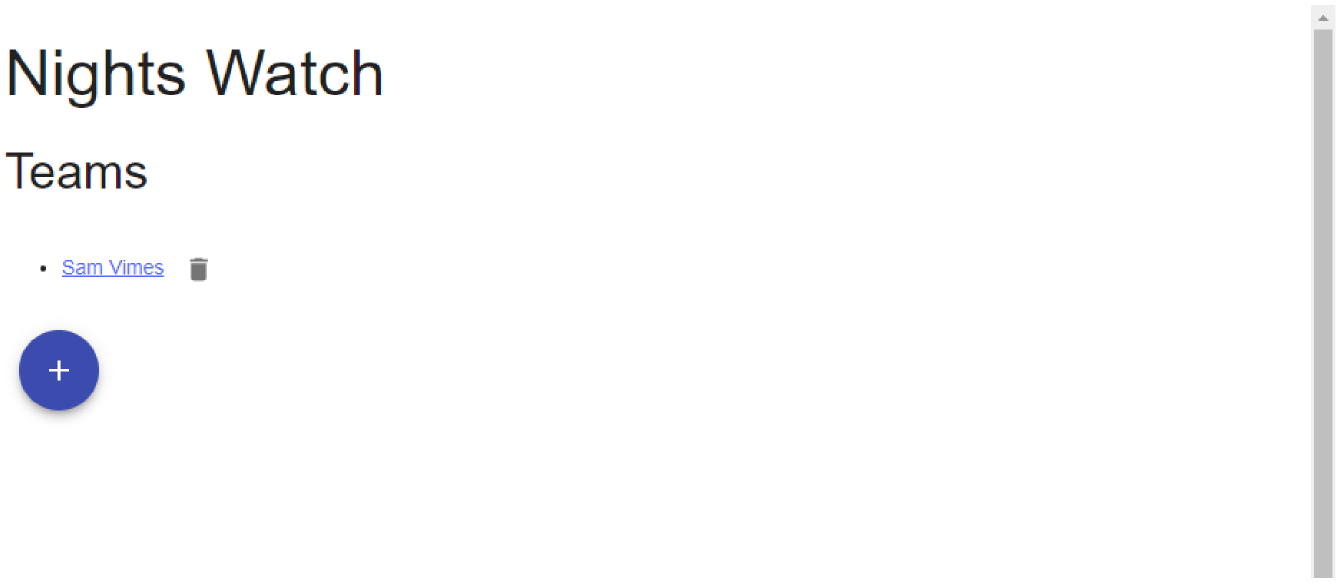


Nights Watch

Teams

- Sam Vimes
- Findthee Swing

6-Deleted swing



Nights Watch

Teams

- Sam Vimes

Test Run No. 1.5

JUnit Tests

com.saskcow.bowling.BowlingApplicationTests

1 tests, 0 failed, 0 errors,

Classname	Name	Time
com.saskcow.bowling.BowlingApplicationTests	contextLoads	0.046
com.saskcow.bowling.BowlingApplicationTests		0.046

com.saskcow.bowling.controller.LeagueControllerTest

3 tests, 0 failed, 0 errors,

Classname	Name	Time
com.saskcow.bowling.controller.LeagueControllerTest	deleteLeague_shouldDeleteLeague	0.22
com.saskcow.bowling.controller.LeagueControllerTest	getLeague_shouldFilter	0.126
com.saskcow.bowling.controller.LeagueControllerTest	addLeague_shouldSaveTheCourse	0.115
com.saskcow.bowling.controller.LeagueControllerTest		0.462

com.saskcow.bowling.controller.PlayerControllerTest

2 tests, 0 failed, 0 errors,

Classname	Name	Time
com.saskcow.bowling.controller.PlayerControllerTest	deleteLeague_shouldDeleteLeague	0.075
com.saskcow.bowling.controller.PlayerControllerTest	addLeague_shouldSaveTheCourse	0.036
com.saskcow.bowling.controller.PlayerControllerTest		0.112

com.saskcow.bowling.controller.TeamControllerTest

2 tests, 0 failed, 0 errors,

Classname	Name	Time
com.saskcow.bowling.controller.TeamControllerTest	deleteLeague_shouldDeleteLeague	0.036
com.saskcow.bowling.controller.TeamControllerTest	addLeague_shouldSaveTheCourse	0.076
com.saskcow.bowling.controller.TeamControllerTest		0.112

com.saskcow.bowling.repository.GameRepositoryTest

1 tests, 0 failed, 0 errors,

Classname	Name	Time
com.saskcow.bowling.repository.GameRepositoryTest	thingsSaved_canBeRetrieved	0.067
com.saskcow.bowling.repository.GameRepositoryTest		0.067

com.saskcow.bowling.repository.LeagueRepositoryTest

3 tests, 0 failed, 0 errors,

Classname	Name	Time
com.saskcow.bowling.repository.LeagueRepositoryTest	thingsSaved_canBeRetrieved	0.011
com.saskcow.bowling.repository.LeagueRepositoryTest	thingsSaved_canBeQueried	0.146

Classname	Name	Time
com.saskcow.bowling.repository.LeagueRepositoryTest	thingsSaved_canBeDeleted	0.032
com.saskcow.bowling.repository.LeagueRepositoryTest		0.191

Nightwatch Tests

Test times not related to how long the site takes to use

TestLeague

Test Results

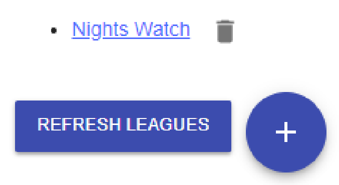
TestLeague

2 tests, 0 failed, 0 errors,

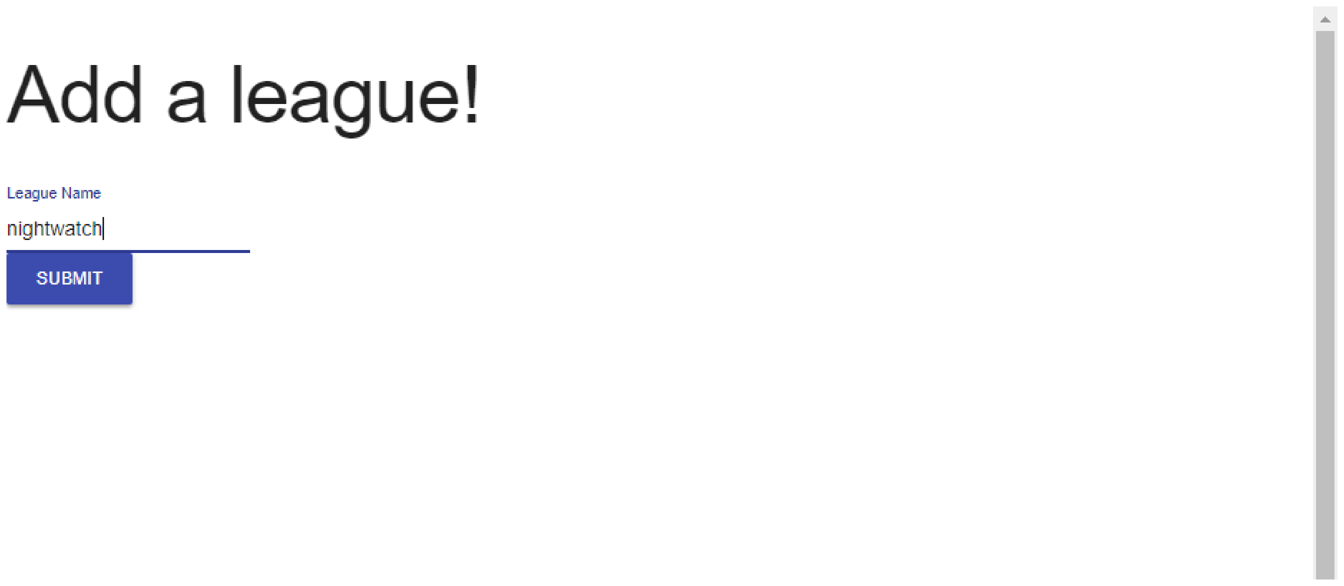
Classname	Name	Time	Assertions
TestLeague	Test adding leagues	7.651	5
TestLeague	Test removing leagues	1.648	3
TestLeague		9.299	

Screenshots

1-start



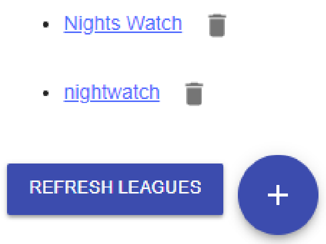
2-adding a league


Add a league!

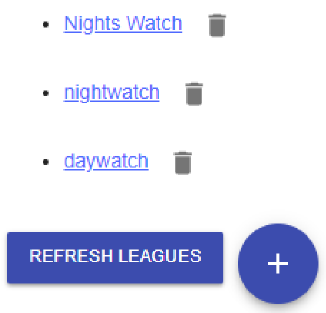
League Name

nightwatch

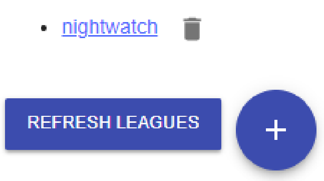
3-Shows League


- Nights Watch
- nightwatch

4-Second League


- Nights Watch
- nightwatch
- daywatch

5-Deleted daywatch



TestTeams

Test Results

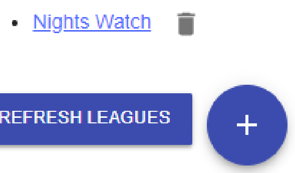
TestTeams

3 tests, 0 failed, 0 errors,

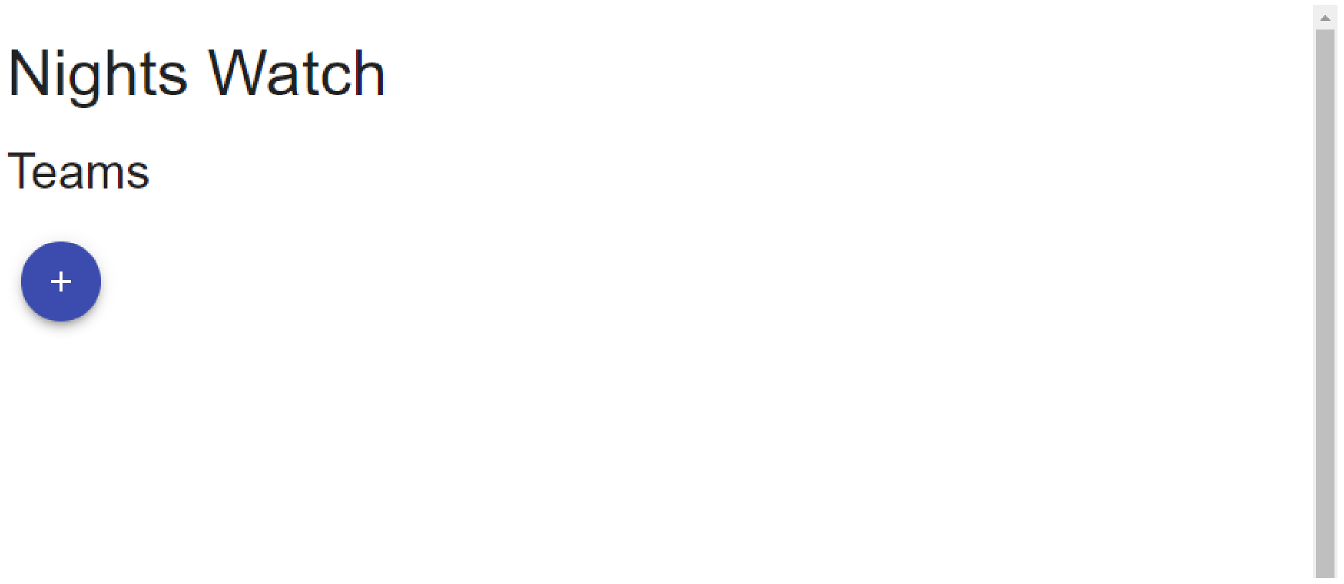
Classname	Name	Time	Assertions
TestTeams	Setup	4.978	3
TestTeams	Test Adding Teams	4.600	6
TestTeams	Deleting	1.786	4
TestTeams		11.36	

Screenshots

1-init league

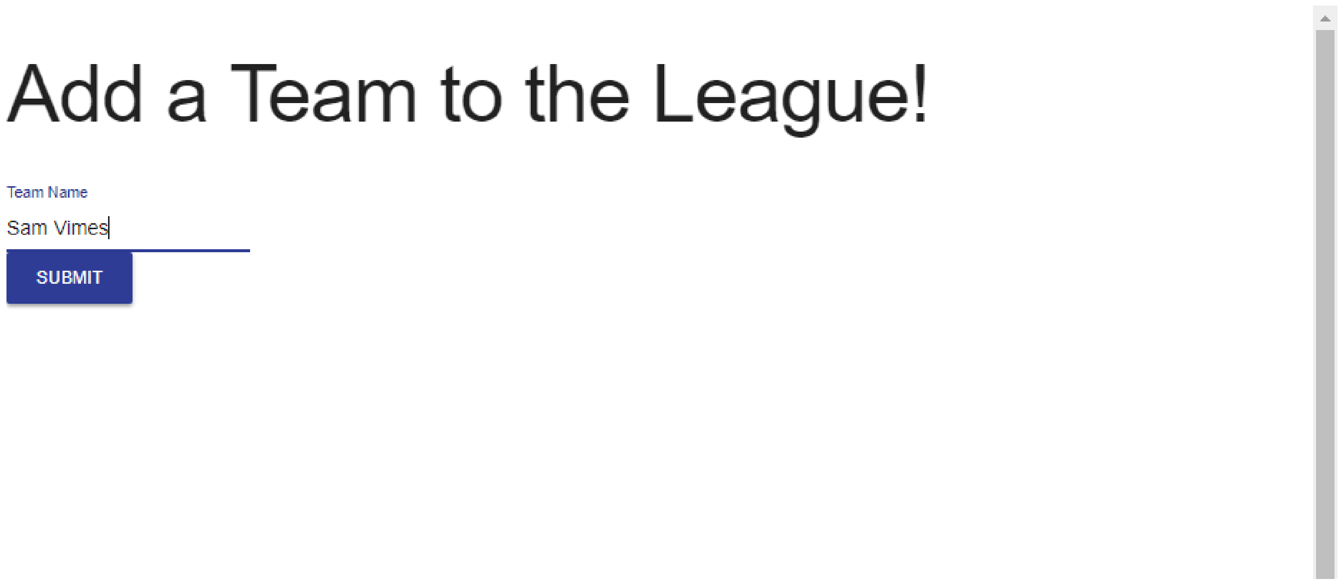


2-League view


Nights Watch

Teams

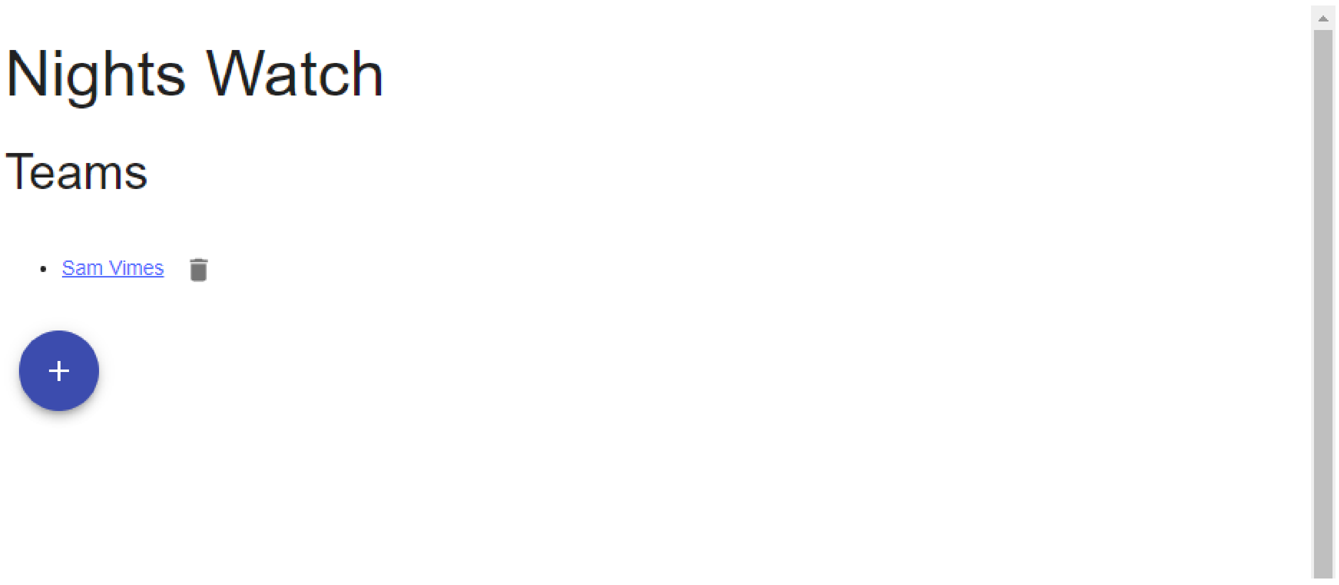
3-Add Team screen


Add a Team to the League!

Team Name

Sam Vimes

4-Sam Vimes in the watch

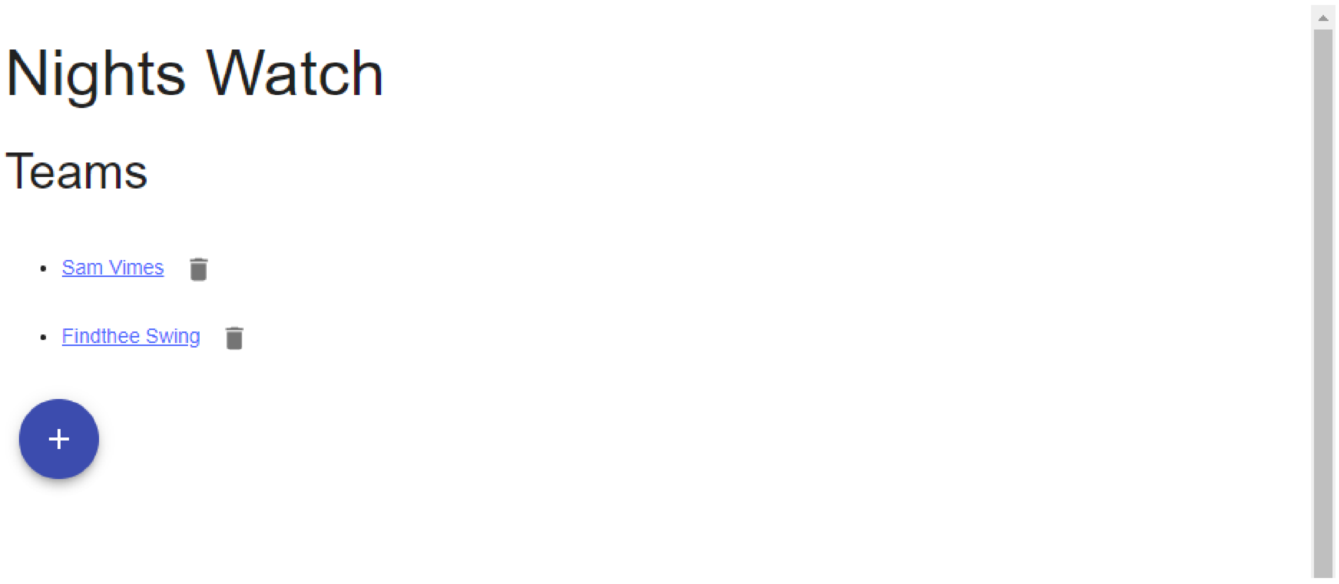

Nights Watch

Teams

- Sam Vimes

+

5-2 teams

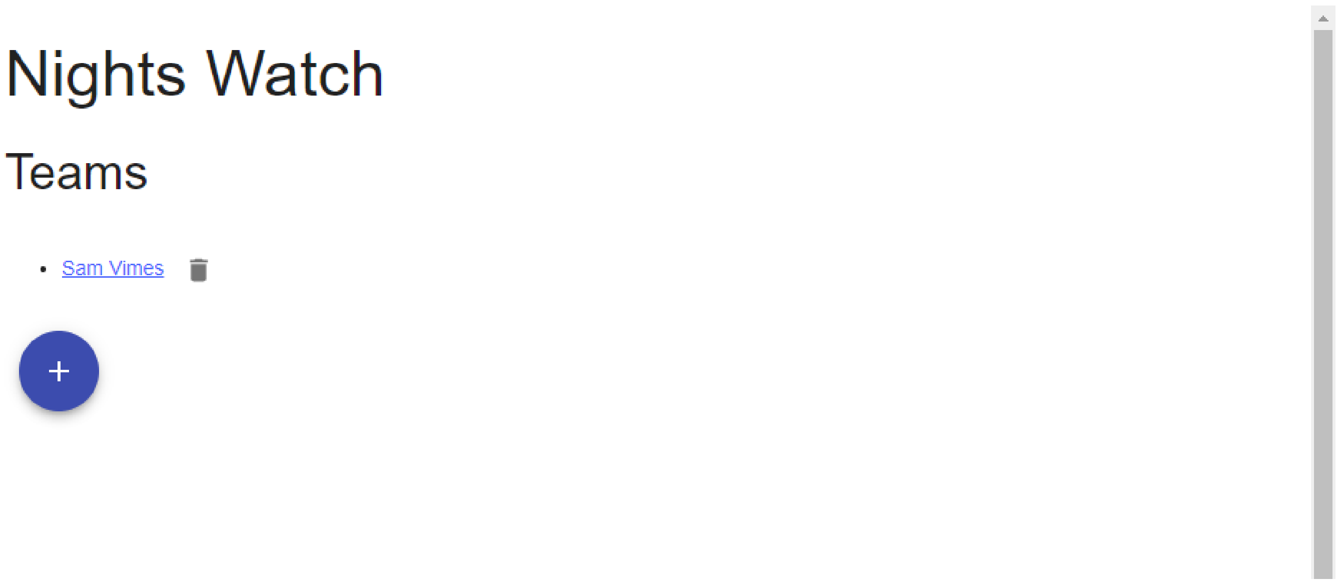

Nights Watch

Teams

- Sam Vimes
- Findthee Swing

+

6-Deleted swing


Nights Watch

Teams

- Sam Vimes

+

Test Run No. 2

JUnit Tests

com.saskcow.bowling.BowlingApplicationTests

1 tests, 0 failed, 0 errors,

Classname	Name	Time
com.saskcow.bowling.BowlingApplicationTests	contextLoads	0.027
com.saskcow.bowling.BowlingApplicationTests		0.027

com.saskcow.bowling.controller.LeagueControllerTest

3 tests, 0 failed, 0 errors,

Classname	Name	Time
com.saskcow.bowling.controller.LeagueControllerTest	deleteLeague_shouldDeleteLeague	0.112
com.saskcow.bowling.controller.LeagueControllerTest	getLeague_shouldFilter	0.069
com.saskcow.bowling.controller.LeagueControllerTest	addLeague_shouldSaveTheCourse	0.068
com.saskcow.bowling.controller.LeagueControllerTest		0.25

com.saskcow.bowling.controller.PlayerControllerTest

2 tests, 0 failed, 0 errors,

Classname	Name	Time
com.saskcow.bowling.controller.PlayerControllerTest	deleteLeague_shouldDeleteLeague	0.07
com.saskcow.bowling.controller.PlayerControllerTest	addLeague_shouldSaveTheCourse	0.022
com.saskcow.bowling.controller.PlayerControllerTest		0.093

com.saskcow.bowling.controller.TeamControllerTest

2 tests, 0 failed, 0 errors,

Classname	Name	Time
com.saskcow.bowling.controller.TeamControllerTest	deleteLeague_shouldDeleteLeague	0.018
com.saskcow.bowling.controller.TeamControllerTest	addLeague_shouldSaveTheCourse	0.042
com.saskcow.bowling.controller.TeamControllerTest		0.06

com.saskcow.bowling.repository.GameRepositoryTest

1 tests, 0 failed, 0 errors,

Classname	Name	Time
com.saskcow.bowling.repository.GameRepositoryTest	thingsSaved_canBeRetrieved	0.097
com.saskcow.bowling.repository.GameRepositoryTest		0.097

com.saskcow.bowling.repository.LeagueRepositoryTest

3 tests, 0 failed, 0 errors,

Classname	Name	Time
com.saskcow.bowling.repository.LeagueRepositoryTest	thingsSaved_canBeRetrieved	0.009
com.saskcow.bowling.repository.LeagueRepositoryTest	thingsSaved_canBeQueried	0.166
com.saskcow.bowling.repository.LeagueRepositoryTest	thingsSaved_canBeDeleted	0.04
com.saskcow.bowling.repository.LeagueRepositoryTest		0.216

Nightwatch Tests

Test times not related to how long the site takes to use

TestLeague

Test Results

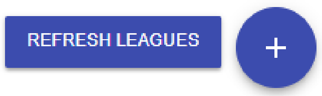
TestLeague

2 tests, 0 failed, 0 errors,

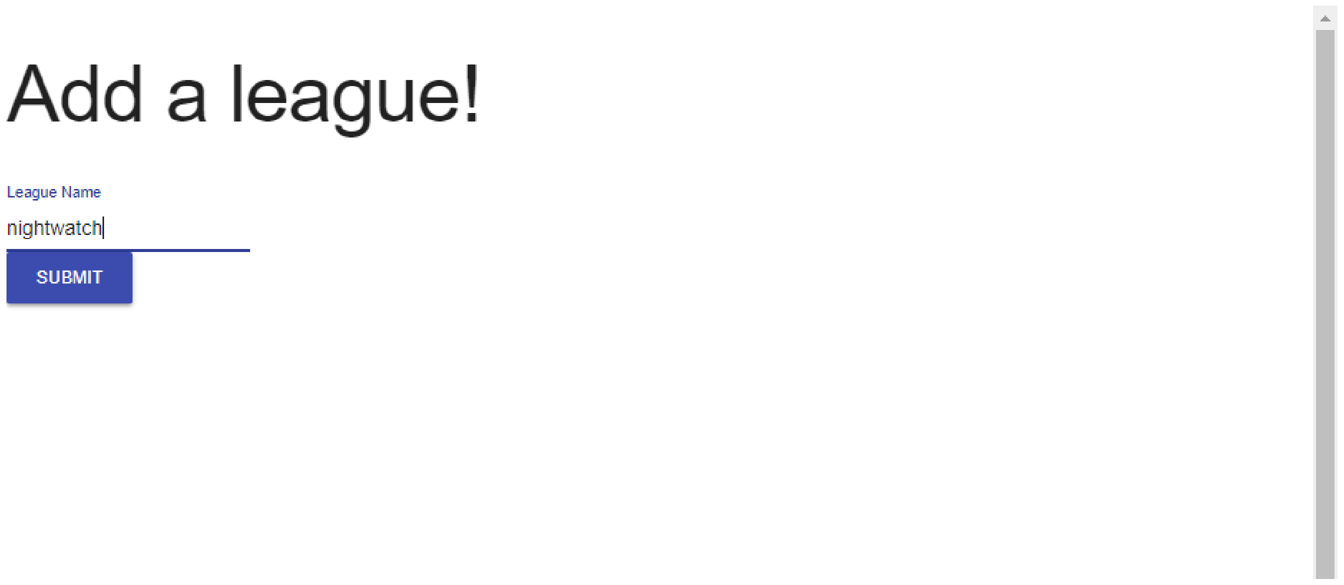
Classname	Name	Time	Assertions
TestLeague	Test adding leagues	7.707	5
TestLeague	Test removing leagues	1.584	3
TestLeague		9.291	

Screenshots

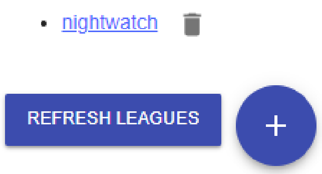
1-start



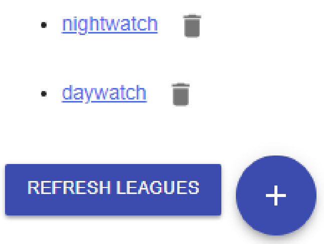
2-adding a league



3-Shows League

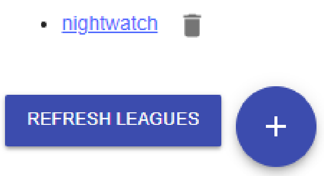


4-Second League


- nightwatch
- daywatch

REFRESH LEAGUES
+

5-Deleted daywatch


- nightwatch

REFRESH LEAGUES
+

TestPlayers

Test Results

TestPlayers

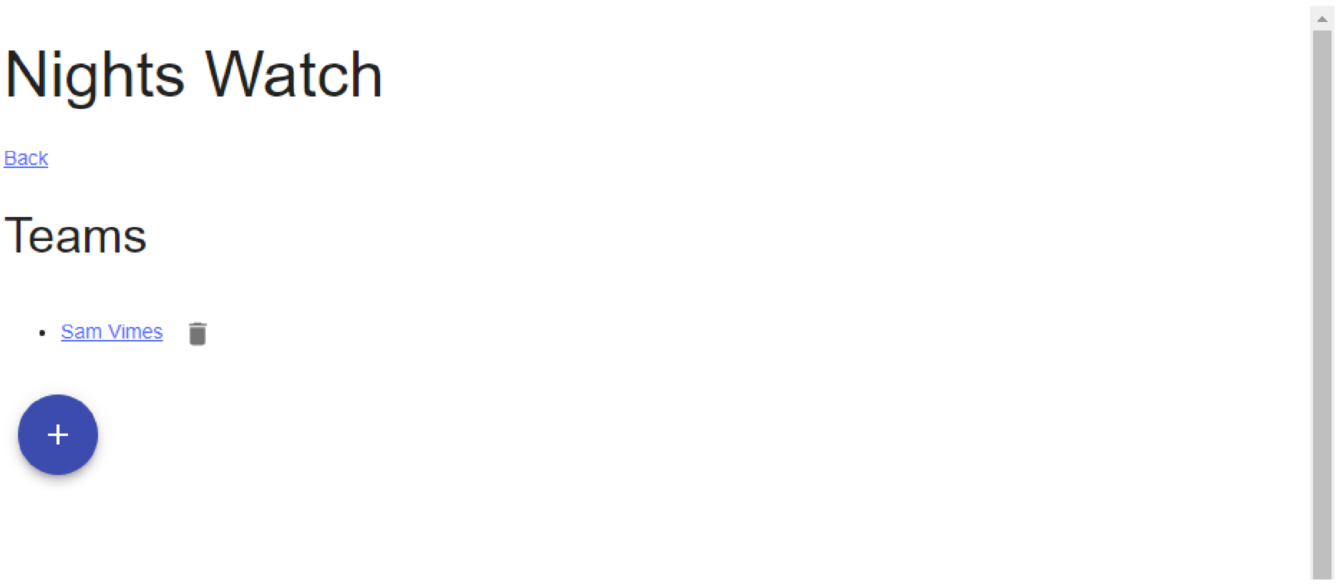
4 tests, 0 failed, 0 errors,

Classname	Name	Time	Assertions
TestPlayers	Setup	7.439	6
TestPlayers	Test Adding Players	4.548	5
TestPlayers	Test Player	1.389	2
TestPlayers	Deleting	1.805	4

Classname	Name	Time	Assertions
TestPlayers		15.18	

Screenshots

1-init team



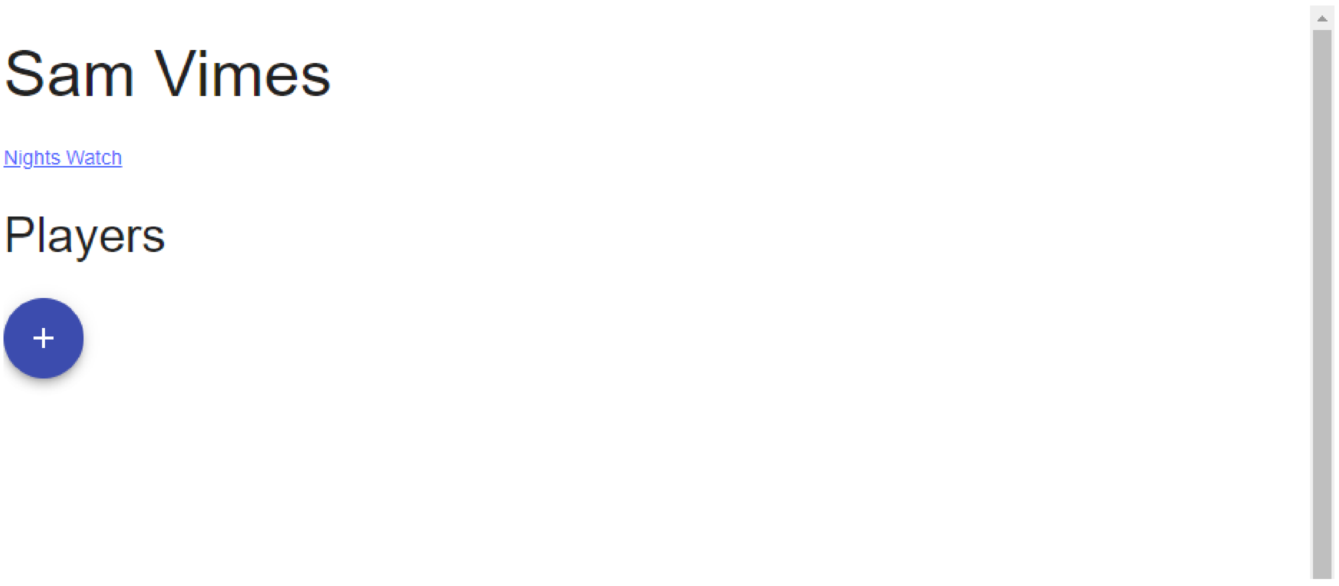
Nights Watch

Back

Teams

- Sam Vimes

2-Team view

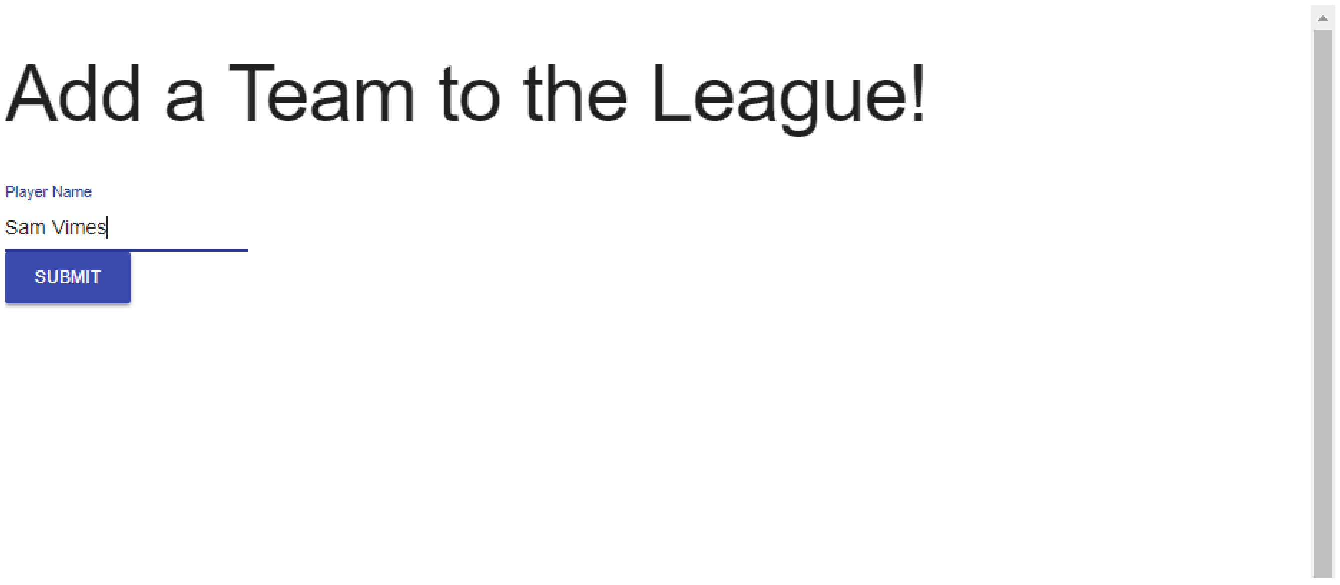


Sam Vimes

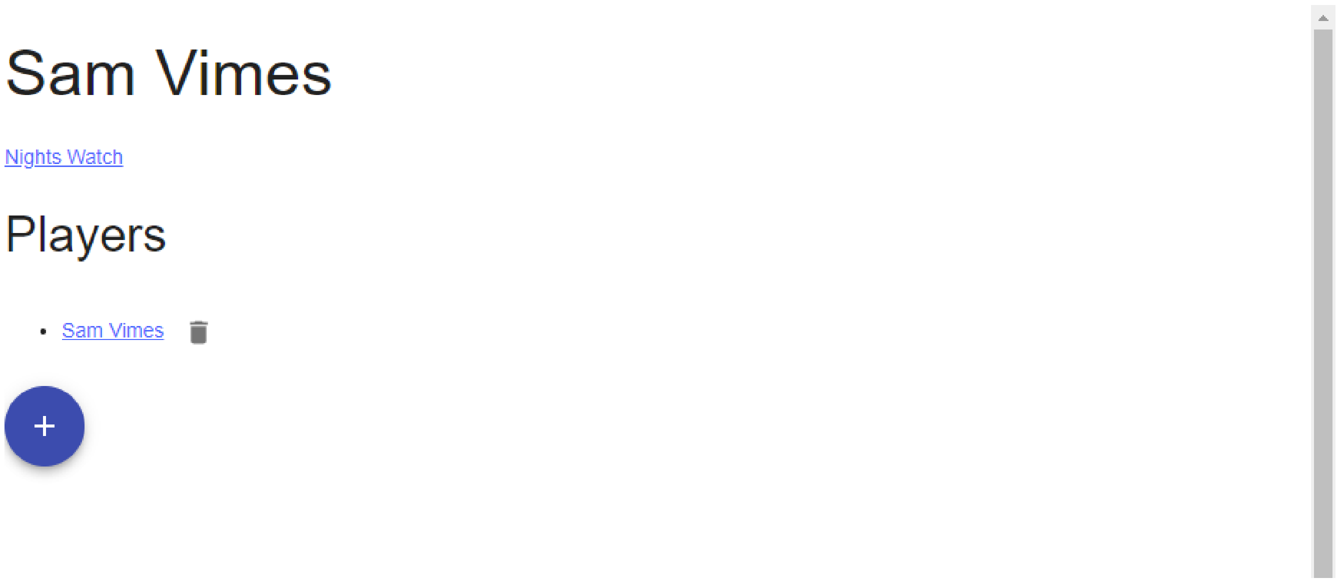
Nights Watch

Players

3-Add Player screen


Add a Team to the League!

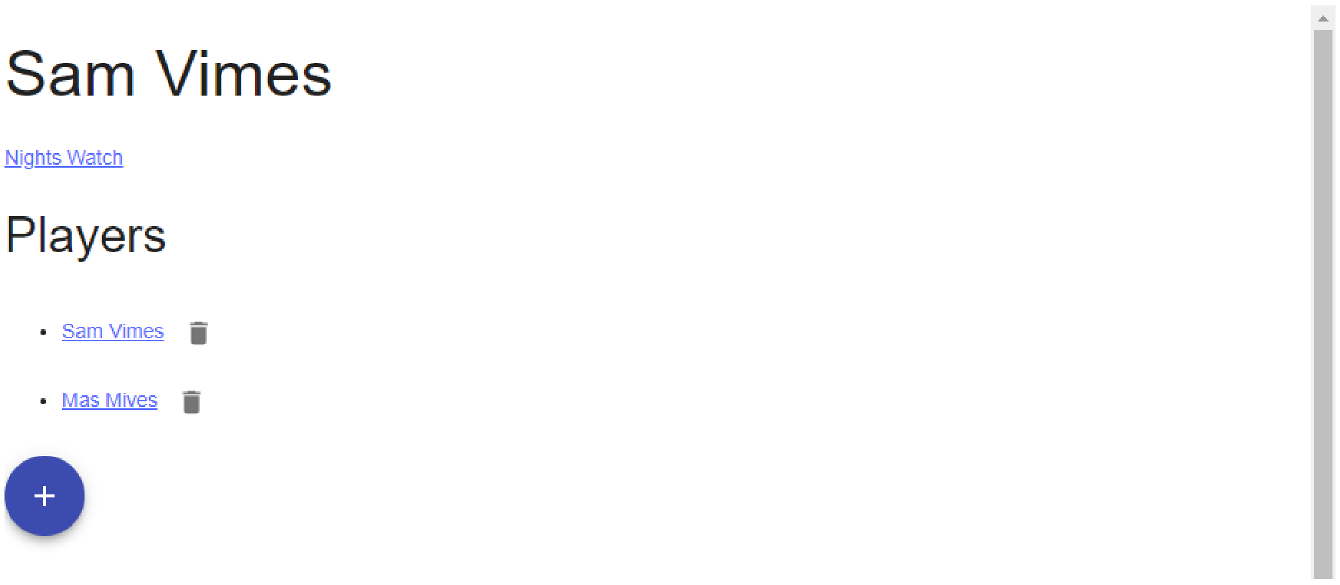
4-Sam Vimes in the Vimes


Sam Vimes

Players

- Sam Vimes

5-2 players


Sam Vimes

Players

- Sam Vimes
- Mas Mives

6-Mas Mives

Mas Mives

Sam Vimes

7-Deleted Mives

Sam Vimes

Nights Watch

Players

- Sam Vimes

TestTeams

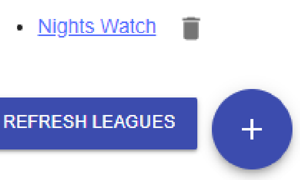
Test Results

TestTeams

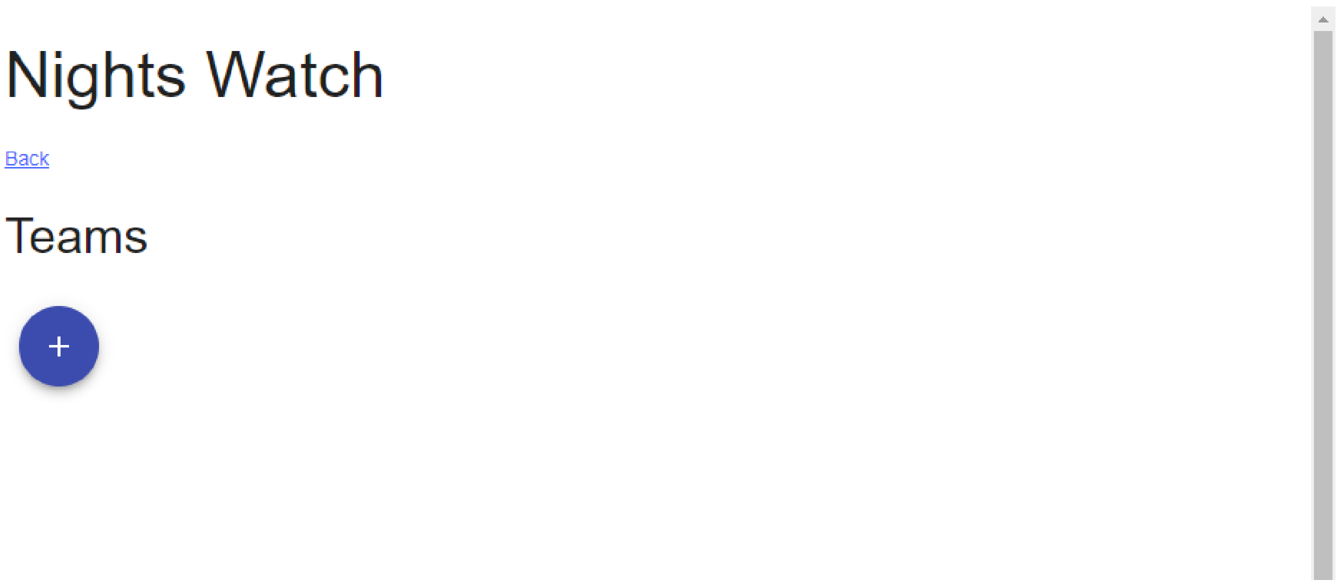
3 tests, 0 failed, 0 errors,

Classname	Name	Time	Assertions
TestTeams	Setup	4.924	3
TestTeams	Test Adding Teams	4.571	6
TestTeams	Deleting	1.790	4
TestTeams		11.29	

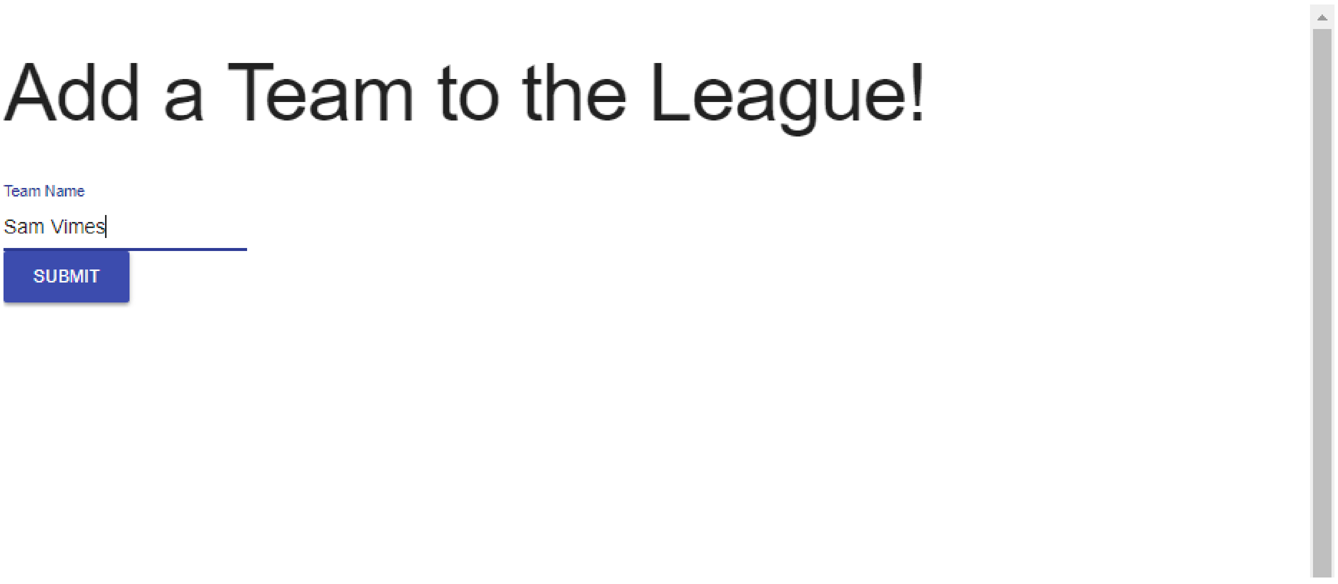
1-init league



2-League view



3-Add Team screen

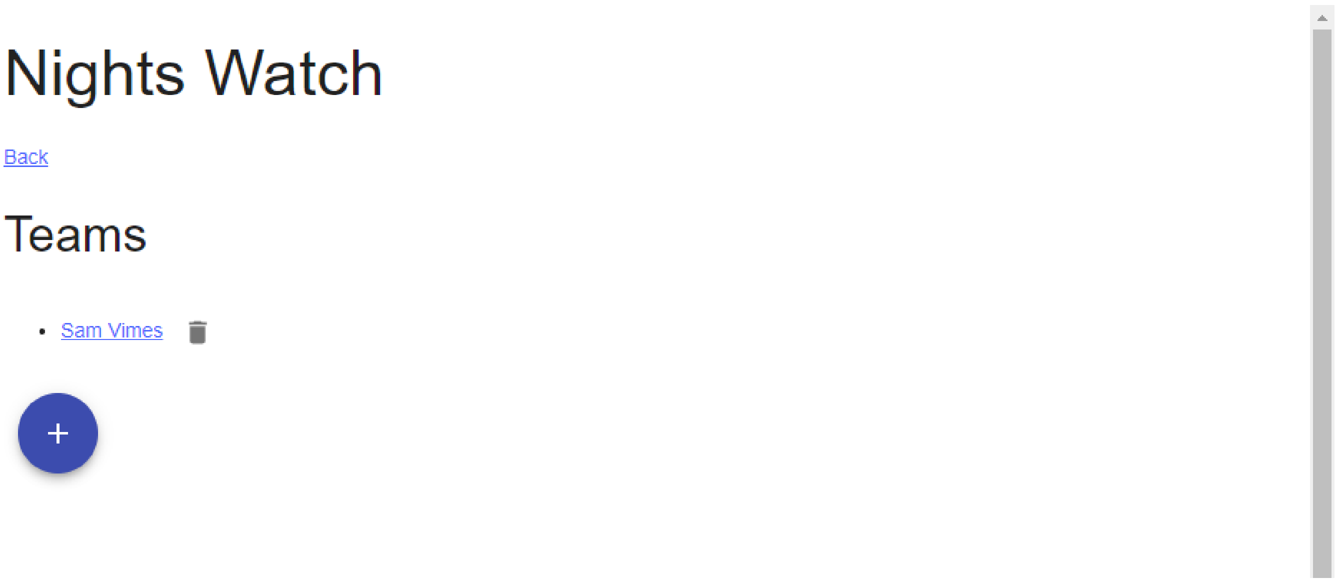

Add a Team to the League!

Team Name

Sam Vimes

SUBMIT

4-Sam Vimes in the watch


Nights Watch

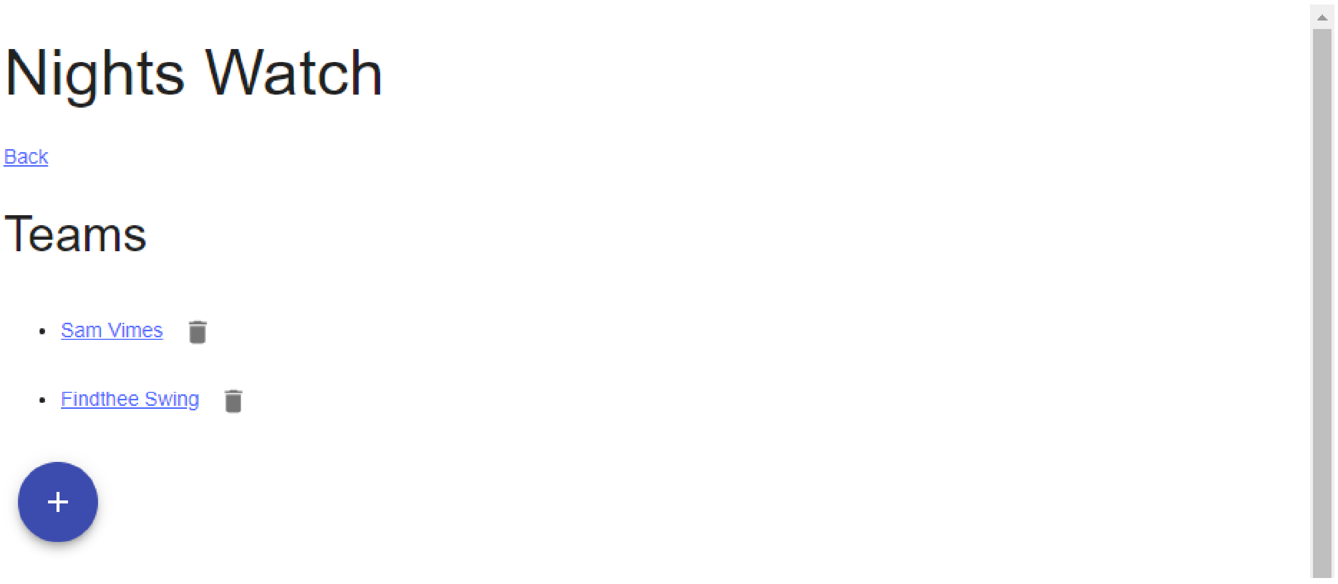
Back

Teams

- Sam Vimes

+

5-2 teams


Nights Watch

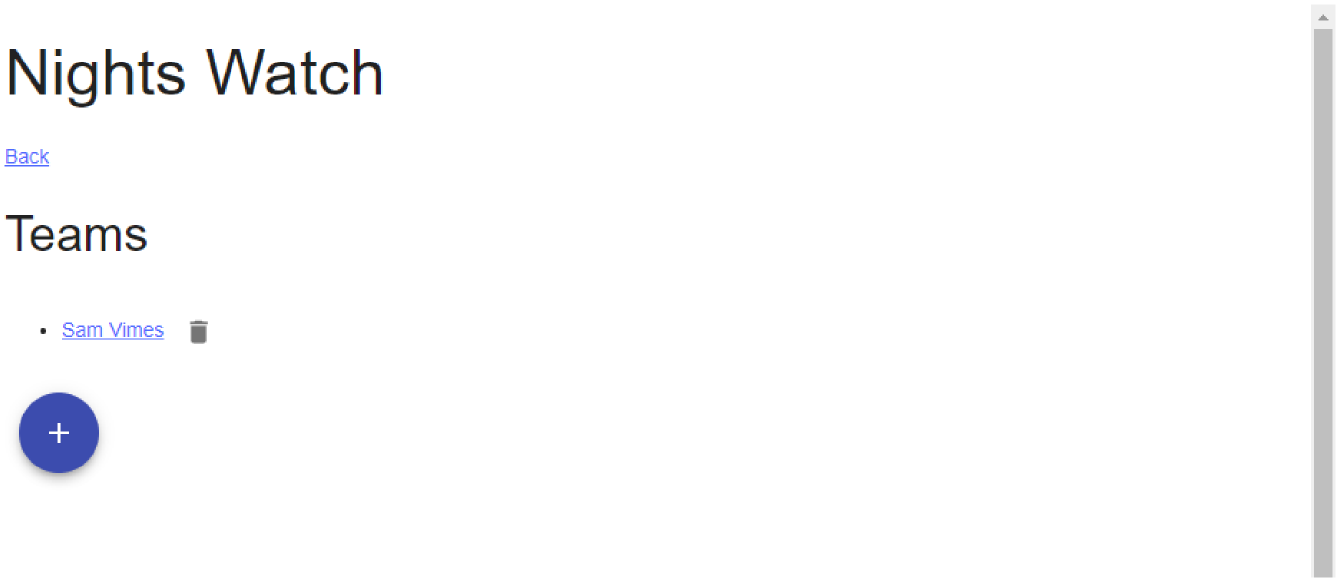
Back

Teams

- Sam Vimes
- Findthee Swing

+

6-Deleted swing


Nights Watch

Back

Teams

- Sam Vimes

Test Run No. 2.5

JUnit Tests

com.saskcow.bowling.BowlingApplicationTests

1 tests, 0 failed, 0 errors,

Classname	Name	Time
com.saskcow.bowling.BowlingApplicationTests	contextLoads	0.049
com.saskcow.bowling.BowlingApplicationTests		0.049

com.saskcow.bowling.controller.GameControllerTest

2 tests, 0 failed, 0 errors,

Classname	Name	Time
com.saskcow.bowling.controller.GameControllerTest	addGame_shouldSaveTheGame	0.322
com.saskcow.bowling.controller.GameControllerTest	deleteGame_shouldDeleteGame	0.165
com.saskcow.bowling.controller.GameControllerTest		0.488

com.saskcow.bowling.controller.LeagueControllerTest

3 tests, 0 failed, 0 errors,

Classname	Name	Time
com.saskcow.bowling.controller.LeagueControllerTest	addLeague_shouldSaveTheLeague	0.097
com.saskcow.bowling.controller.LeagueControllerTest	deleteLeague_shouldDeleteLeague	0.03

Classname	Name	Time
com.saskcow.bowling.controller.LeagueControllerTest	getLeague_shouldFilter	0.049
com.saskcow.bowling.controller.LeagueControllerTest		0.177

com.saskcow.bowling.controller.PlayerControllerTest

2 tests, 0 failed, 0 errors,

Classname	Name	Time
com.saskcow.bowling.controller.PlayerControllerTest	deletePlayer_shouldDeletePlayer	0.031
com.saskcow.bowling.controller.PlayerControllerTest	addPlayer_shouldSaveThePlayer	0.057
com.saskcow.bowling.controller.PlayerControllerTest		0.089

com.saskcow.bowling.controller.TeamControllerTest

2 tests, 0 failed, 0 errors,

Classname	Name	Time
com.saskcow.bowling.controller.TeamControllerTest	addTeam_shouldSaveTheTeam	0.05
com.saskcow.bowling.controller.TeamControllerTest	deleteTeam_shouldDeleteTeam	0.029
com.saskcow.bowling.controller.TeamControllerTest		0.08

com.saskcow.bowling.repository.GameRepositoryTest

1 tests, 0 failed, 0 errors,

Classname	Name	Time
com.saskcow.bowling.repository.GameRepositoryTest	thingsSaved_canBeRetrieved	0.087
com.saskcow.bowling.repository.GameRepositoryTest		0.087

com.saskcow.bowling.repository.LeagueRepositoryTest

3 tests, 0 failed, 0 errors,

Classname	Name	Time
com.saskcow.bowling.repository.LeagueRepositoryTest	thingsSaved_canBeRetrieved	0.013
com.saskcow.bowling.repository.LeagueRepositoryTest	thingsSaved_canBeQueried	0.204
com.saskcow.bowling.repository.LeagueRepositoryTest	thingsSaved_canBeDeleted	0.039
com.saskcow.bowling.repository.LeagueRepositoryTest		0.259

Nightwatch Tests

Test times not related to how long the site takes to use

TestLeague

Test Results

TestLeague

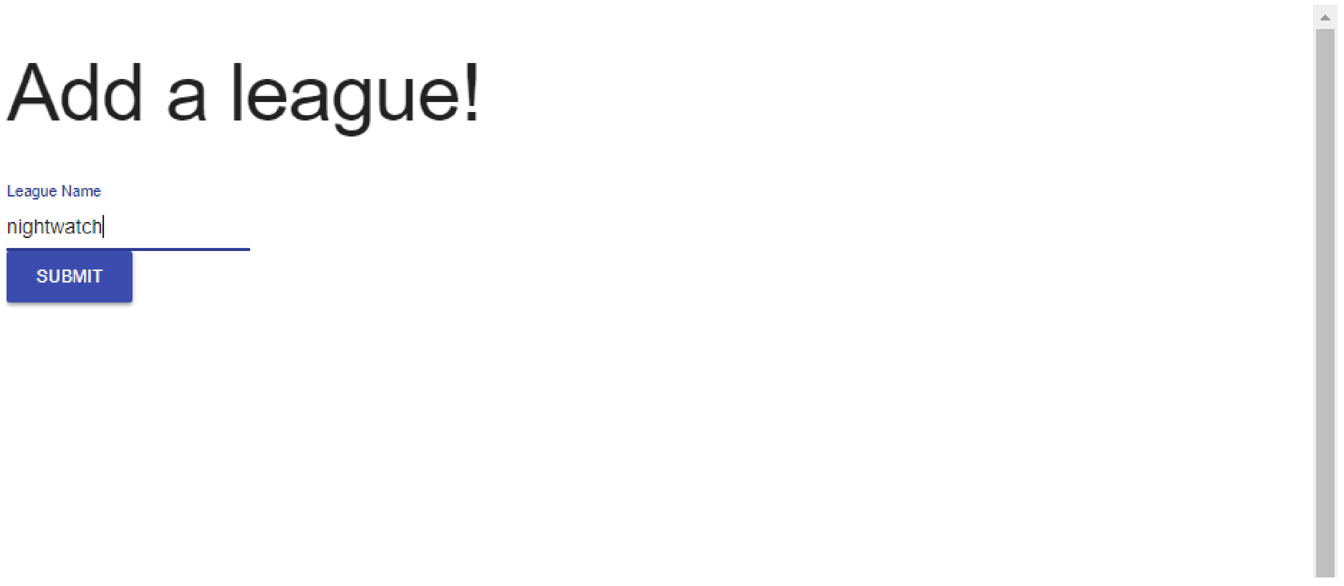
2 tests, 0 failed, 0 errors,

Classname	Name	Time	Assertions
TestLeague	Test adding leagues	7.707	5
TestLeague	Test removing leagues	1.584	3
TestLeague		9.291	

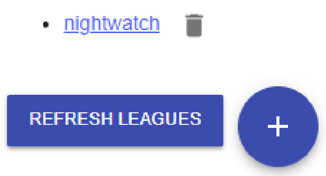
Screenshots

1-start

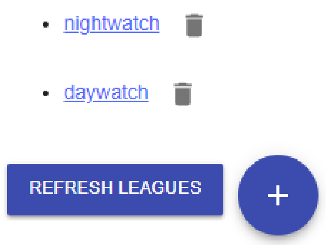
2-adding a league



3-Shows League



4-Second League



5-Deleted daywatch

TestPlayers

Test Results

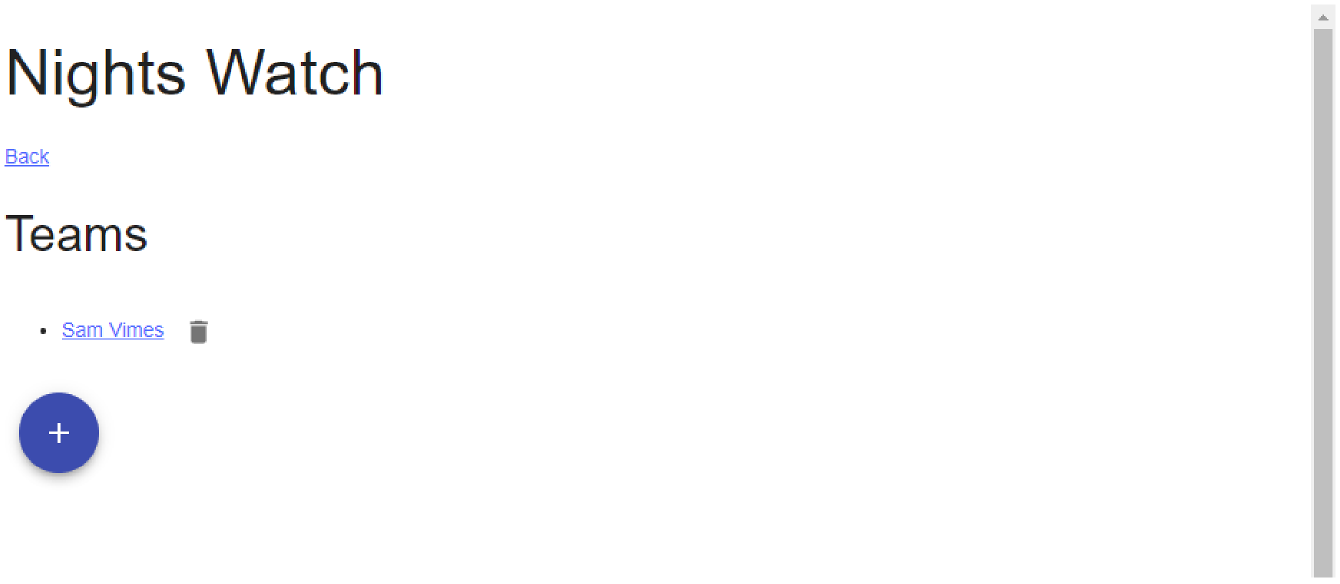
TestPlayers

4 tests, 0 failed, 0 errors,

Classname	Name	Time	Assertions
TestPlayers	Setup	7.439	6
TestPlayers	Test Adding Players	4.548	5
TestPlayers	Test Player	1.389	2
TestPlayers	Deleting	1.805	4
TestPlayers		15.18	

Screenshots

1-init team


Nights Watch

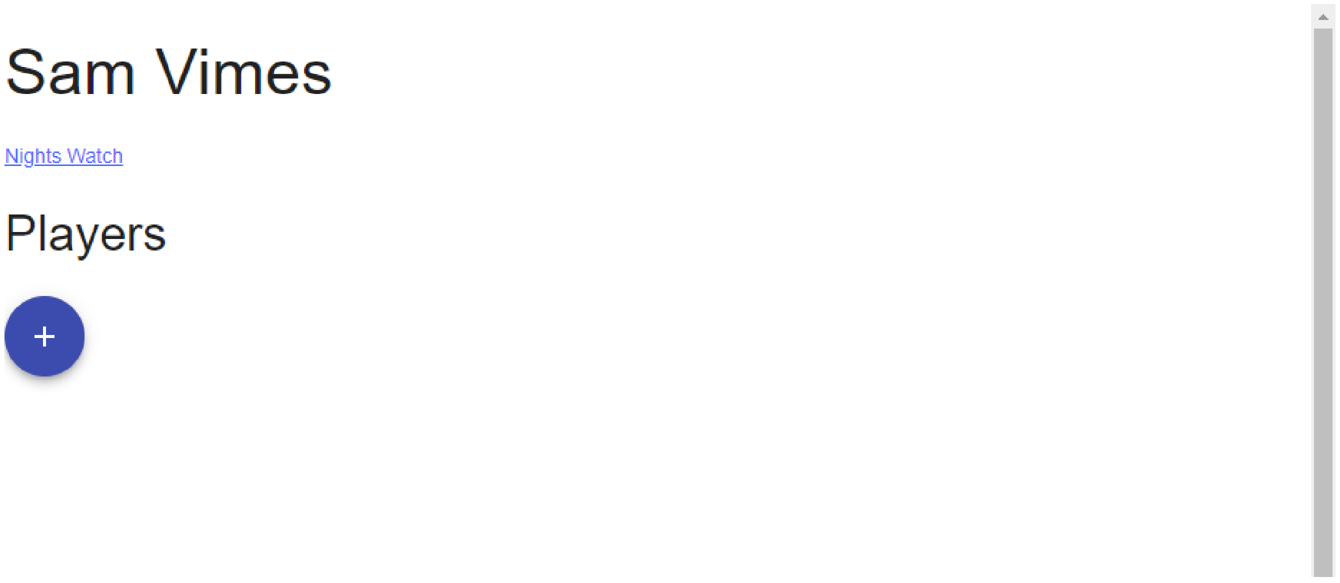
Back

Teams

- Sam Vimes

+

2-Team view

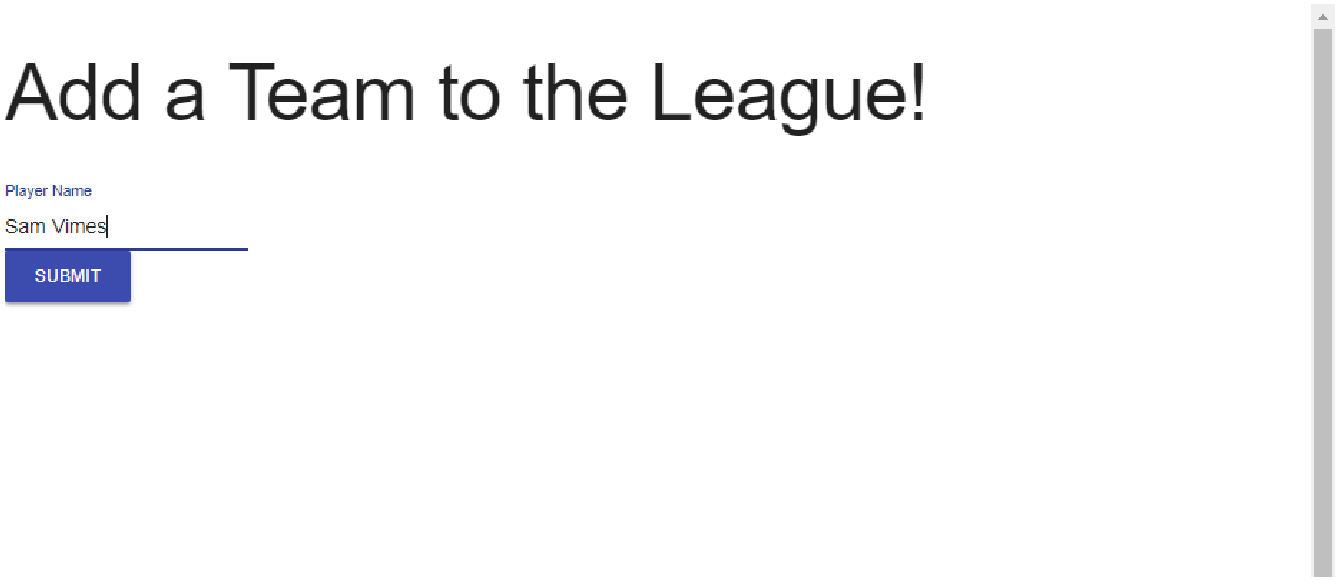

Sam Vimes

Nights Watch

Players

+

3-Add Player screen

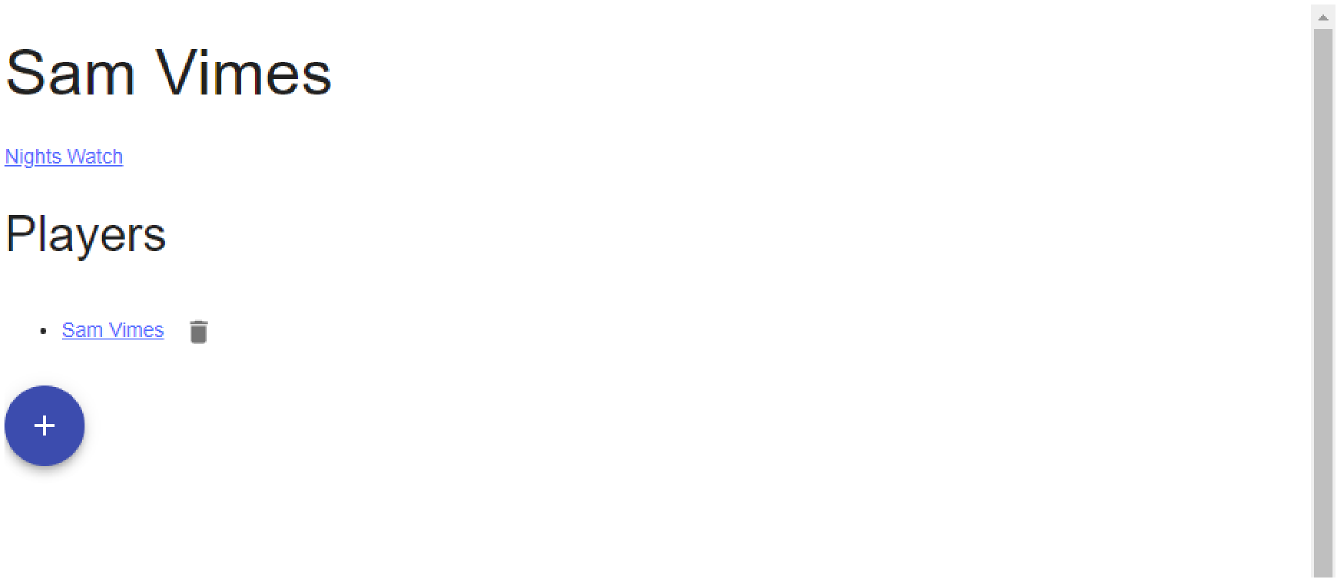

Add a Team to the League!

Player Name

Sam Vimes

SUBMIT

4-Sam Vimes in the Vimes

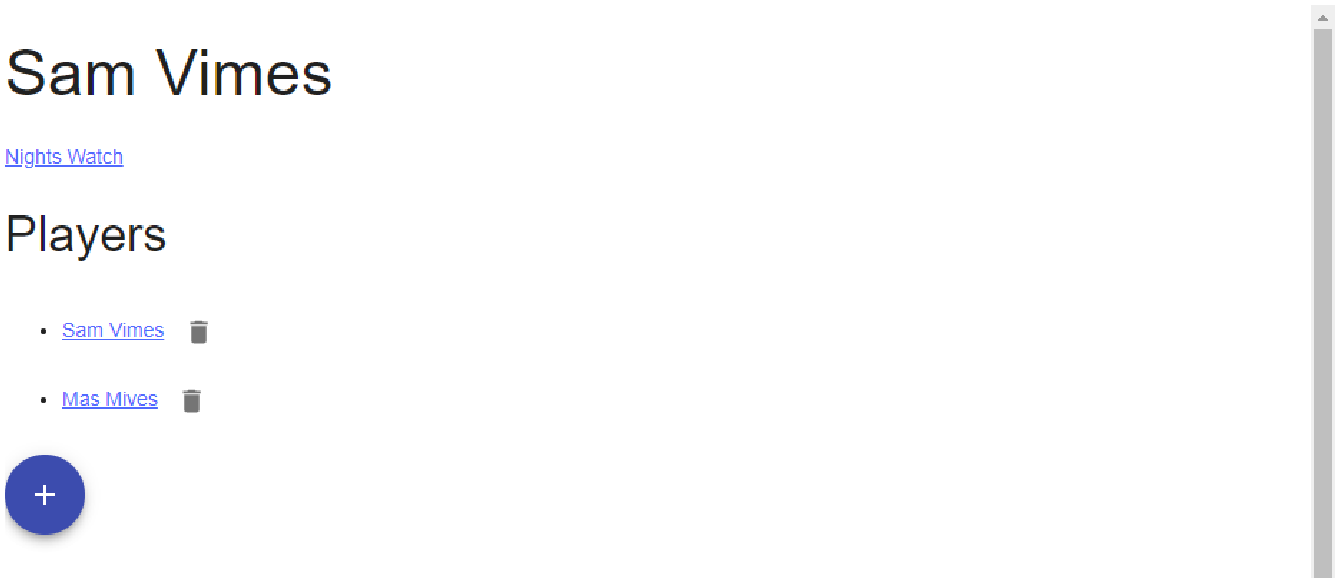

Sam Vimes

Nights Watch

Players

- Sam Vimes
-

5-2 players


Sam Vimes

Nights Watch

Players

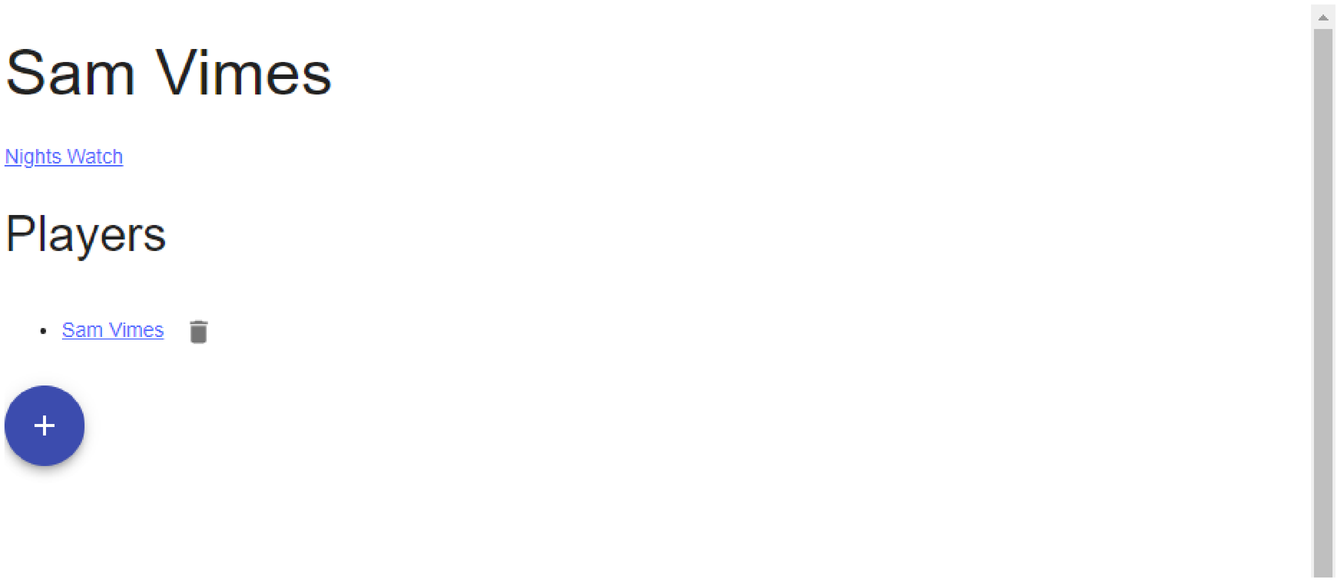
- Sam Vimes
- Mas Mives

6-Mas Mives

Mas Mives

Sam Vimes

7-Deleted Mives


Sam Vimes

Nights Watch

Players

- Sam Vimes

TestTeams

Test Results

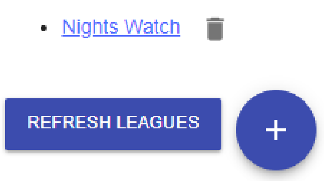
TestTeams

3 tests, 0 failed, 0 errors,

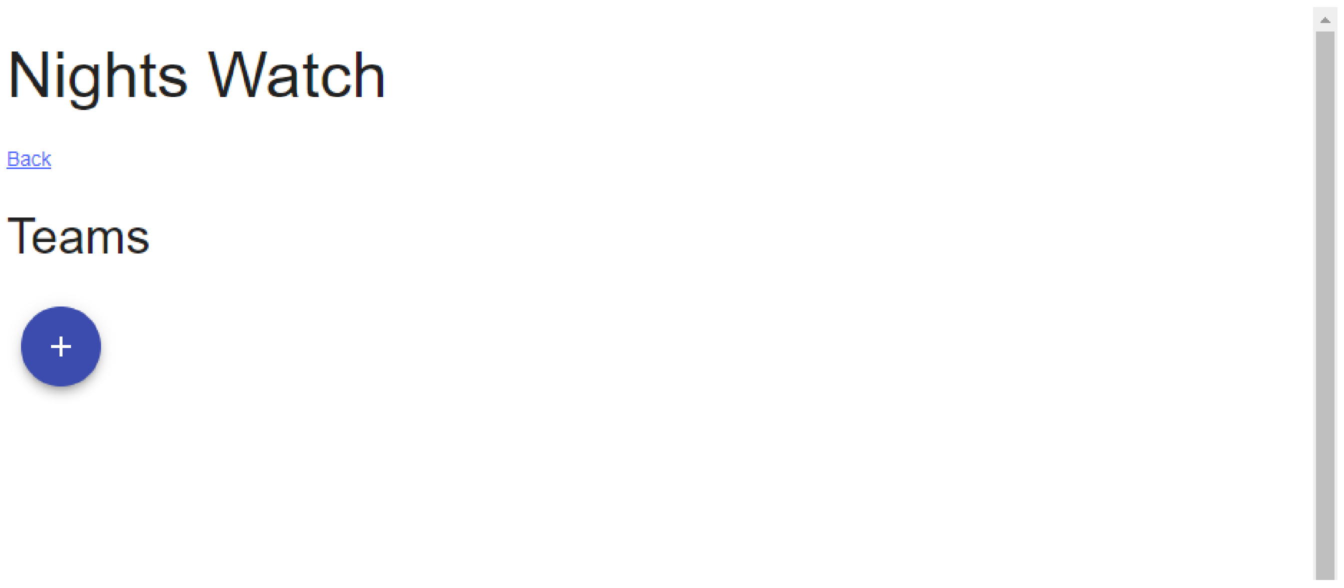
Classname	Name	Time	Assertions
TestTeams	Setup	4.924	3
TestTeams	Test Adding Teams	4.571	6
TestTeams	Deleting	1.790	4
TestTeams		11.29	

Screenshots

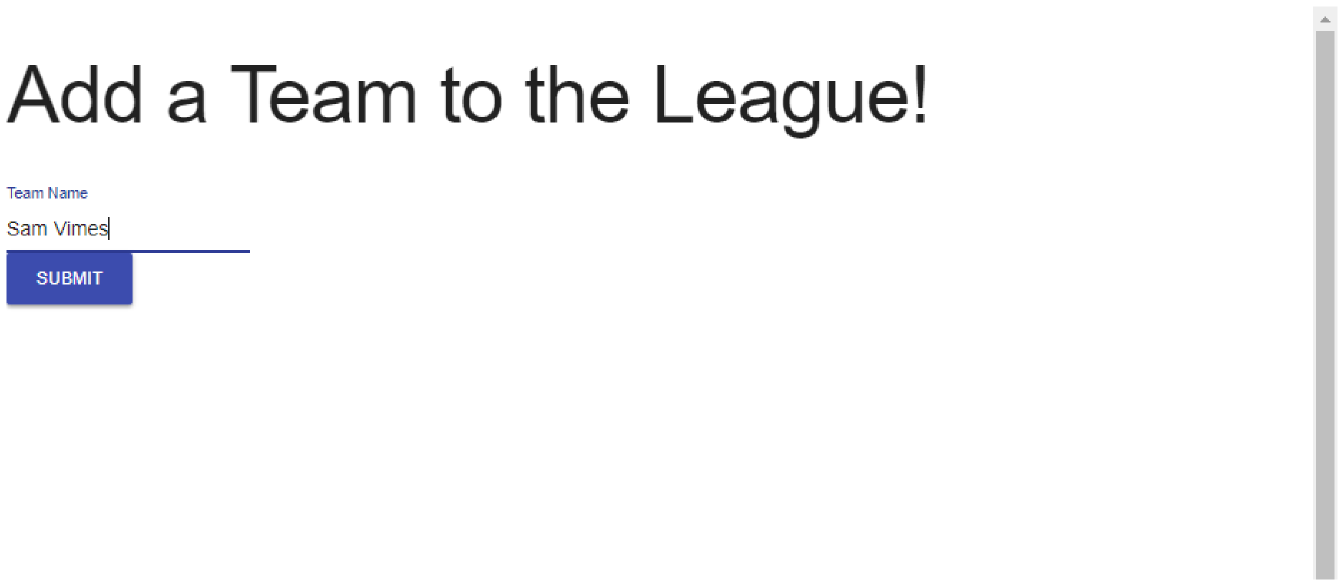
1-init league



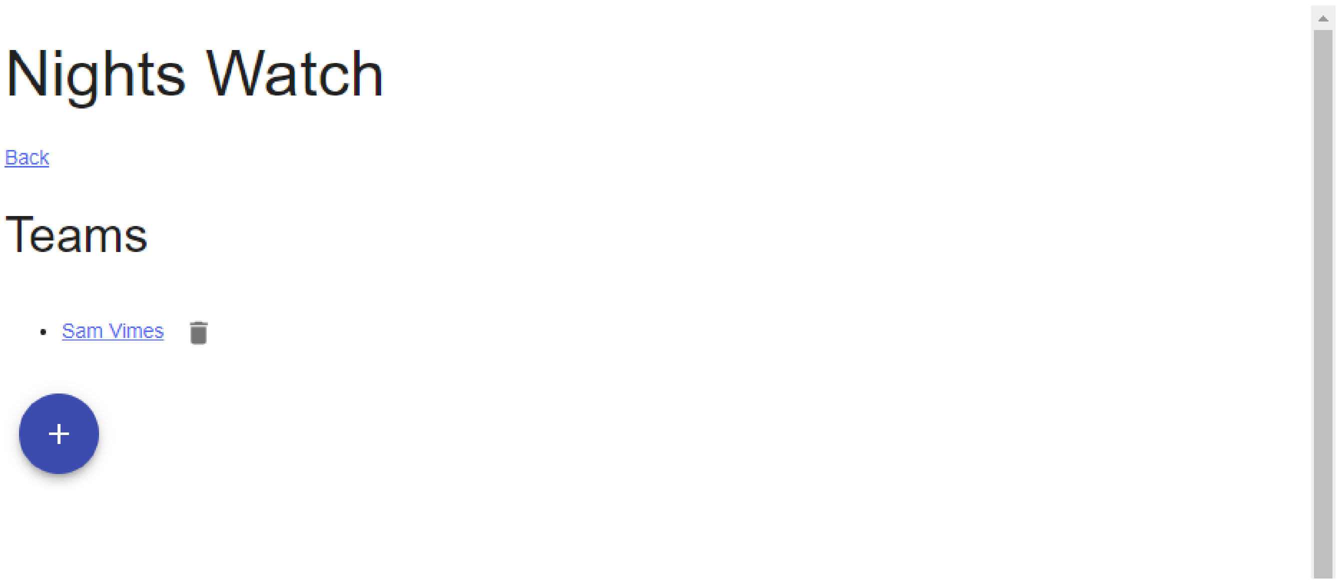
2-League view



3-Add Team screen



4-Sam Vimes in the watch

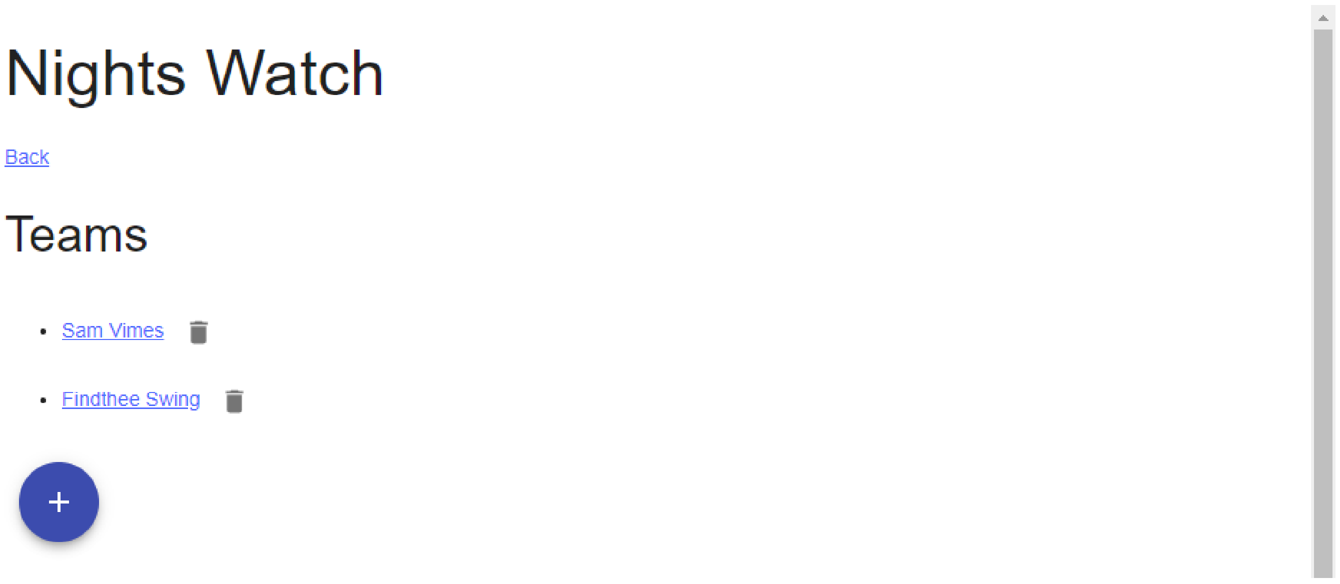

Nights Watch

Back

Teams

- Sam Vimes

5-2 teams

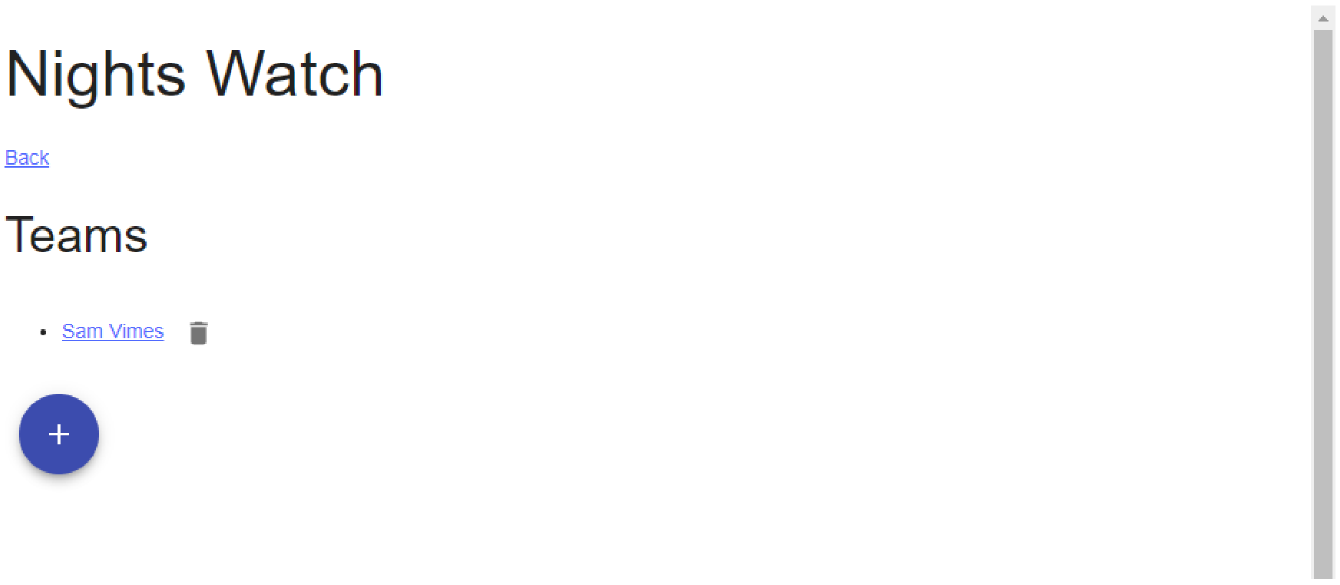

Nights Watch

Back

Teams

- Sam Vimes
- Findthee Swing

6-Deleted swing


Nights Watch

Back

Teams

- Sam Vimes

6. Api Docs

General API stuff

6.1. League API

6.1.1. Create A League

```
POST /api/league HTTP/1.1
Content-Type: application/json
Host: localhost:8080
Content-Length: 16

{"name":"Brian"}
```

```
HTTP/1.1 201 Created
Location: http://localhost:8080/api/league/1
```

Object sent usually will just be a name to create a league off, and all other objects will be created off of league.

6.1.2. Get a list of Leagues

```
GET /api/league HTTP/1.1
Host: localhost:8080
```

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 25

[{"id":1,"name":"Brian"}]
```

Get’s a summary of ALL leagues, returns as a list of "LeagueViewSummary" objects

6.1.3. Get one League

```
GET /api/league/1 HTTP/1.1
Host: localhost:8080
```

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 45

{"id":1,"name":"Brian","teams":[],"games":[]}
```

Gets one league based off of it’s id. Single League gives more detail, returns a "LeagueView" not a "LeagueViewSummary"

6.1.4. Get a list of leagues by name

```
GET /api/league?name=Bri HTTP/1.1
Host: localhost:8080
```

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 25

[{"id":3,"name":"Brian"}]
```

The same as getting a list of all leagues, but will only show leagues with matching criteria. Matches off contains not exact match.

6.1.5. Delete A league

```
DELETE /api/league/1 HTTP/1.1
Host: localhost:8080
```

```
HTTP/1.1 204 No Content
```

Also removes any teams associated with the league.

6.2. Team API

The Team API is simpler than the league API due to a "LeagueView" returning a list of "TeamViewSummary" already, this makes the use of a list redundant for the application.

6.2.1. Create A Team

```
POST /api/team HTTP/1.1
Content-Type: application/json
Host: localhost:8080
Content-Length: 33

{"name":"Brian", "leagueId": "1"}
```

```
HTTP/1.1 201 Created
Location: http://localhost:8080/api/team/1
```

The team creation takes a name for the team and a leagueid, to create a team with the league.

6.2.2. Get A Team

```
GET /api/team/1 HTTP/1.1
Host: localhost:8080
```

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 84

{"league":{"id":1,"name":"Brian"},"id":1,"name":"Brian","players":null,"games":null}
```

This returns one team based off of ID, a list is not necessary due to a league giving a list.
Returns a "TeamView"

6.2.3. Delete A Team

```
DELETE /api/team/1 HTTP/1.1
Host: localhost:8080
```

```
HTTP/1.1 204 No Content
```

Deletes the team, does not delete if the team has games.

6.3. Player API

Similar to team API as a team gives a list of players

6.3.1. Create A Player

```
POST /api/player HTTP/1.1
Content-Type: application/json
Host: localhost:8080
Content-Length: 31

{"name":"Brian", "teamId": "1"}
```

```
HTTP/1.1 201 Created
Location: http://localhost:8080/api/player/1
```

The player creation takes a name for the player and a teamid, to create a player with the team.

6.3.2. Get A Player

```
GET /api/player/1 HTTP/1.1
Host: localhost:8080
```

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 65

{"id":1,"team":{"id":1,"name":"Brian's Bowlers!"},"name":"Brian"}
```

This returns one player based off of ID, a list is not necessary due to a team giving a list.
Returns a "PlayerView"

6.3.3. Delete A Player

```
DELETE /api/player/1 HTTP/1.1
Host: localhost:8080
```

```
HTTP/1.1 204 No Content
```

Deletes the player.

6.4. Team API

The Team API is simpler than the league API due to a "LeagueView" returning a list of "TeamViewSummary" already, this makes the use of a list redundant for the application.

6.4.1. Create A Team

```
POST /api/team HTTP/1.1
Content-Type: application/json
Host: localhost:8080
Content-Length: 33

{"name":"Brian", "leagueId": "1"}
```

```
HTTP/1.1 201 Created
Location: http://localhost:8080/api/team/1
```

The team creation takes a name for the team and a leagueid, to create a team with the league.

6.4.2. Get A Team

```
GET /api/team/1 HTTP/1.1
Host: localhost:8080
```

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 84

{"league":{"id":1,"name":"Brian"},"id":1,"name":"Brian","players":null,"games":null}
```

This returns one team based off of ID, a list is not necessary due to a league giving a list.
Returns a "TeamView"

6.4.3. Delete A Team

```
DELETE /api/team/1 HTTP/1.1
Host: localhost:8080
```

```
HTTP/1.1 204 No Content
```

Deletes the team, does not delete if the team has games.

6.5. Game API

Games between 2 teams, at a given venue and time

6.5.1. Create A Game

```
POST /api/game HTTP/1.1
Content-Type: application/json
Host: localhost:8080
Content-Length: 99

{"time":"2018-02-18T12:22:54.174", "venue": "Brian Bowling Centre", "teamId1": "1",
"teamId2": "2"}
```

```
HTTP/1.1 201 Created
Location: http://localhost:8080/api/game/1
```

The game creation takes a time (ISO local datetime string) and a venue (string) as well as ids for both teams involved in the game, upcoming games are not currently supported

6.5.2. Get A Game

```
GET /api/game/1 HTTP/1.1
Host: localhost:8080
```

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 162

{"id":1,"time":"2018-02-18T12:22:54.174","teams":[{"id":1,"name":"Dave"},{"id":2
,"name":"David"}],"venue":"Brian Bowling Centre","league":{"id":1,"name":"Brian"}}
```

This returns one game based off of ID, a list is not necessary due to a team giving a list.
Returns a "GameView" should rarely be used as a league and a team shows games as "GameView"

6.5.3. Delete A Game

```
DELETE /api/game/1 HTTP/1.1
Host: localhost:8080
```

```
HTTP/1.1 204 No Content
```

Deletes the game.