

Gymnastics Scoring Software

The purpose of this paper is to give a detailed overview of our current progress and remaining work to be done.

10.0 Software

Progress Report

CS 462

Senior Software Engineering Project

Winter 2019

Prepared for
Michael Chaplin

Prepared by
Nicholas Giles
Mary Jacobsen
Thomas Huynh
Zech DeCleene

Abstract

This document outlines the progress made by 10.0 Software and provides an overview of the system as well as documents the design and functionality of the alpha and beta builds.

Contents

1	Recap	2
1.1	System Purpose	2
1.2	System Goals	2
1.3	System Overview	2
2	Current Progress	2
2.1	Alpha	2
3	Beta	4
4	Problems	5
4.1	Database	5
4.2	Sessions	5
4.2.1	Creating the meet table	5
4.2.2	Creating the score table	5
5	User Study	5
6	Application	6
6.1	Create Meet page	6
6.2	Lineup page	7
6.3	Score page	8

1 Recap

1.1 System Purpose

The system we will be designing is for the scoring of gymnastics meets held in Gill Coliseum. The system will take in, calculate, organize, format, and display scores as well as export and print the scores.

1.2 System Goals

The purpose of this system is to provide an improved viewing experience for OSU Gymnastics spectators as well as make the scoring process easier for the judges. The system will be comprised of a web application running on the Gill servers that is accessible by the meet staff to input or adjust scores. The scores will then be formatted to be displayed on the scoreboards inside Gill. The application will be able to format all current data into a score sheet that the user can print out at any time. The system will also comprise of a back-end API with a SQL database running on the Gill servers to store all scoring information.

1.3 System Overview

The system is comprised of three main components.

- **Web Application:** The Web application's main function will be the user interface which allows the meet staff to add, edit and print scores. The web application will be responsible for formatting and calculating scores then pushing the data to the database and scoreboard.
- **Scoreboard Utilization:** Data collected from the web application will be pushed to display up-to-date information on current events of the meet.
- **Database:** The database will maintain current and past information. The information will include the teams, judges, athletes, and the scores associated with each athlete and judge.

2 Current Progress

Currently we are in the process of improving our alpha version and getting it ready for beta. Improvements include changing the structure of the database, client side JavaScript, printing, and security.

2.1 Alpha

The alpha version of the software has these features currently implemented:

1. Docker containerization
2. Working MySQL database
Initially 3 separate files
3. Working endpoints for /team, /player and /lineup

```
--Example of player-init.sql--
```

```
DROP TABLE IF EXISTS `player`;  
CREATE TABLE `player` (  
  `id` mediumint(9) NOT NULL AUTO_INCREMENT,  
  `playerNum` mediumint(9),  
  `name` varchar(255) NOT NULL,  
  `team` varchar(255) NOT NULL,  
  `vaultScore` DECIMAL(13,10),  
  `barsScore` DECIMAL(13,10),  
  `beamScore` DECIMAL(13,10),  
  `floorScore` DECIMAL(13,10),
```

```

`AAScore` DECIMAL(13,10),
PRIMARY KEY (`id`),
KEY `idx_name` (`name`)
) ENGINE=InnoDB AUTO_INCREMENT=20 DEFAULT CHARSET=latin1;

```

4. Functional scoring and lineup pages

//--Example of client-side request--

```

function getTeamData() {
  var url = window.location.origin;
  $.getJSON(url + "/team/teams", (data) => {
    var teams = [];
    console.log(data)
    $.each(data, (key, val) => {
      teams.push("<option value='" +
        val.teamName + "'"> " +
        val.teamName + "</option>");
    });

    $("#team-select").append(teams.join(""));
  });
}

```

//--Example of server-side response--

```

router.get('/:id', function (req, res, next) {
  console.log(" -- req.params:", req.params.id);
  const mysqlPool = req.app.locals.mysqlPool;
  const id = req.params.id;
  getTeamByID(id, mysqlPool)
    .then((id) => {
      if (id) {
        res.status(200).json(id);
      } else {
        next();
      }
    })
    .catch((err) => {
      res.status(500).json({
        error: "Unable to fetch team. Please try again later."
      });
    });
});

function getTeamByID(id, mysqlPool) {
  return new Promise((resolve, reject) => {
    mysqlPool.query('SELECT * FROM team WHERE id = ?', [ id ], function (err, results) {
      if (err) {
        reject(err);
      } else {
        resolve(results[0]);
      }
    });
  });
};

```

3 Beta

For beta we have the following things implemented.

1. Newly structured database

Single db-init file

Use of foreign keys to relate all tables to meetID

Example meet table

```
CREATE TABLE `player` (  
  `id` mediumint(9) NOT NULL AUTO_INCREMENT,  
  `playerNum` mediumint(9),  
  `name` varchar(255) NOT NULL,  
  `teamID` mediumint(9) NOT NULL,  
  `vaultScore` DECIMAL(13,10),  
  `barsScore` DECIMAL(13,10),  
  `beamScore` DECIMAL(13,10),  
  `floorScore` DECIMAL(13,10),  
  `AAScore` DECIMAL(13,10),  
  `meetID` mediumint(9),  
  PRIMARY KEY (`id`),  
  KEY `fk_team_player` (`teamID`),  
  KEY `fk_meet_player` (`meetID`),  
  
  CONSTRAINT `team_fk_1`  
  FOREIGN KEY (`teamID`)  
  REFERENCES `team`(`id`)  
  ON UPDATE CASCADE  
  ON DELETE NO ACTION,  
  
  CONSTRAINT `meet_fk_2`  
  FOREIGN KEY (meetID)  
  REFERENCES meet(id)  
  ON UPDATE CASCADE  
  ON DELETE NO ACTION  
) ENGINE=InnoDB AUTO_INCREMENT=20 DEFAULT CHARSET=latin1;
```

2. API works with all endpoints:

/Meet

/Team

/Playe

/Judge

/Lineup

/Score

3. JQuery/ajax on all pages

4. Security implementation

INCLUDE EXAMPLE CODE HERE

5. HTML/CSS completed for all pages

6. Cookies that stores meet ID

INCLUDE EXAMPLE CODE HERE

7. Printing capability(in progress)

8. Export CSV for Road to Nationals(in progress)

4 Problems

4.1 Database

The biggest problem we encountered Winter term was having to reconfigure the database. As we were testing our alpha version of the software, we found errors that made it necessary to look into how our database was organized. The errors were that there was not an easy way to organize the data into separate meets and that scores were not associated with a judge. The reason we need scores associated with a judge is because each score on the scoring sheet needs to be listed with the judge that gave it. The reasoning for having the data organized into separate meets is apparent for many reasons, the most important of which, is because scoring sheets need to be accessible for each individual meet. We are fixing these errors by adding a meet object that has all the objects of a meet associated with it as well as a score object that has one player and one judge associated with it. We encountered problems with having to add and update endpoints based on the changed database that slowed progress down.

Reconfiguring the database was completed in week 8 and all the endpoints were added and tested by week 9. With all the endpoints added, the meet setup page could be connected to the back-end. This also allowed cookies to be set up so the current meet id could be stored. Adding a score object that is associated with a player and a judge allows the judges to be included on the score sheet with each score they give. The relational way the database is set up and all the endpoints that were added makes this software scalable because the endpoints can be used for any new features or updates in the future.

4.2 Sessions

After testing beta, we found that we need to add sessions so that a new device can go into the web app and find the current meet id. This would essentially be back-end cookies. We currently have front-end cookies implemented so one device can know the meet id. Sessions would allow the web app to be used by all the devices at each event during a meet.

4.2.1 Creating the meet table

```
CREATE TABLE `meet` (  
  `id` mediumint(9) NOT NULL AUTO_INCREMENT,  
  `name` varchar(255),  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO_INCREMENT=20 DEFAULT CHARSET=latin1;
```

4.2.2 Creating the score table

```
CREATE TABLE `score` (  
  `id` mediumint(9) NOT NULL AUTO_INCREMENT,  
  `playerID` mediumint(9) NOT NULL,  
  `judgeID` mediumint(9) NOT NULL,  
  `score` DECIMAL(13,10) NOT NULL,  
  `event` varchar(255) NOT NULL,  
  `exhibition` BIT NOT NULL,  
  `meetID` mediumint(9) NOT NULL,  
  PRIMARY KEY (`id`),  
  FOREIGN KEY (`playerID`) REFERENCES `player`(`id`),  
  FOREIGN KEY (`judgeID`) REFERENCES `judge`(`id`),  
  FOREIGN KEY (`meetID`) REFERENCES `meet`(`id`)  
) ENGINE=InnoDB AUTO_INCREMENT=20 DEFAULT CHARSET=latin1;
```

5 User Study

We met with our client and completed our first user study on Michael Chaplin (associate head coach of OSU gymnastics). He tested out our alpha software during our last client meeting. During our user study, we had him choose a team and score each member on the team for the vault event. Overall, he found the software relatively simple and easy to use. We did notice a couple of small problems with the software such as the web application lacking a team successfully chosen page. This made Michael click the accept button twice resulting in him having

to score each member of the OSU gymnastics team twice for the vault event. One concern Michael mentioned was that coaches will need the ability to make last second lineup changes in the event of an injury or other extreme circumstance right before an event starts. Overall, Michael Chaplin was satisfied with our progress on the project.

6 Application

6.1 Create Meet page

Scoring Software 10.0

Meet Lineup Print Score

Number of Teams:

1

Accept

Team Names

OSU

Accept

Number of Gymnasts for OSU

2

Gymnasts for OSU

Mary Jacobsen

Nikola Jokic

Gymnasts for OSU

Mary Jacobsen

Nikola Jokic

Accept

Number of Judges:

2

Accept

Judge Names

Michael

John

Accept

Please double check all the information is correct before submitting.

Submit Data

This page is the "Create Meet" page. The user enters the number of teams, team names, gymnast names for each

team, number of judges, and judge names. There is an accept button for every input field because team names, gymnast names, and judge names all depend on number of teams, number of gymnasts, and number of judges respectively. The create meet page must be filled out in order to use the lineup, score, and print pages.

6.2 Lineup page

Pick event to set lineup

Vault

Pick team to select vault lineup

OSU

OSU

UO

Edit OSU lineup:

1: Mary Mary

2: Choose gymnast

3: Choose gymnast

4: Choose gymnast

5: Choose gymnast

6: Choose gymnast

Accept

Scoring Software 10.0

Meet Lineup Print Score

Vault lineup for OSU has been made. Full screen Stop

This is the lineup page. First, the user selects which event to set the lineups for. Then, the user chooses the team to set the lineups for. Finally, the user can set the lineup of that team for the chosen event. After the lineup is set, the user will see a confirmation page.

6.3 Score page

Pick event to score

Vault

Pick team to score for vault

OSU

Pick which judge is scoring vault

Michael

Mary

10.0	9.95	9.9	9.85	9.8	9.75	9.7	9.65	9.6	9.55	9.5	9.45	9.4	9.35	9.3	9.25	9.2	9.15	9.1	9.05	9.0	8.95
8.9	8.85	8.8	8.75	8.7	8.65	8.6	8.55	8.5	8.45	8.4	8.35	8.3	8.25	8.2	8.15	8.1	8.05	8.0	7.95	7.9	7.85
7.8	7.75	7.7	7.65	7.6	7.55	7.5	7.45	7.4	7.35	7.3	7.25	7.2	7.15	7.1	7.05	7.0	6.95	6.9	6.85	6.8	6.75
6.7	6.65	6.6	6.55	6.5	6.45	6.4	6.35	6.3	6.25	6.2	6.15	6.1	6.05	6.0	5.95	5.9	5.85	5.8	5.75	5.7	5.65
5.6	5.55	5.5	5.45	5.4	5.35	5.3	5.25	5.2	5.15	5.1	5.05	5.0	4.95	4.9	4.85	4.8	4.75	4.7	4.65	4.6	4.55
4.5	4.45	4.4	4.35	4.3	4.25	4.2	4.15	4.1	4.05	4.0	3.95	3.9	3.85	3.8	3.75	3.7	3.65	3.6	3.55	3.5	3.45
3.4	3.35	3.3	3.25	3.2	3.15	3.1	3.05	3.0	2.95	2.9	2.85	2.8	2.75	2.7	2.65	2.6	2.55	2.5	2.45	2.4	2.35
2.3	2.25	2.2	2.15	2.1	2.05	2.0	1.95	1.9	1.85	1.8	1.75	1.7	1.65	1.6	1.55	1.5	1.45	1.4	1.35	1.3	1.25
1.2	1.15	1.1	1.05	1.0	0.95	0.9	0.85	0.8	0.75	0.7	0.65	0.6	0.55	0.5	0.45	0.4	0.35	0.3	0.25	0.2	0.15
0.1	0.05	0.0																			

Accept

Vault scoring for OSU has been completed