

University of Arkansas – CSCE Department

Capstone II – Final Report – Fall 2022

“JB Hunt - Intern Hub”

Cayla Johnson, Jessi Soto, Benjamin Thiele, William Jackson

Abstract

Our goal is to implement a web application that assists in the management of interns at the internship program at JB Hunt. In a way, it could be described as an in-house LinkedIn website with less emphasis on social media components. Rather, its primary purpose is to be a convenient resource used to promote/sell current interns to full-time teams.

Each intern will be provided with an editable account. These accounts will provide elements such as a profile picture, basic information, current team, assets in possession, work tasks completed, skillset, long-term goals, and of course, a standard resume. Additional features will include a search/filtering system and possibly a notification system to flag students who are encountering roadblocks in the application.

1.0 Problem

JB Hunt internship sessions can consist of over 100 interns, especially during the summer. All students come from various education levels, experience levels, and locations and have a range of career goals. Typically, all the metrics listed above can be managed through traditional or external means. For example, resumes could be stored as a set of documents in a custom folder, and an excel spreadsheet could easily be used to record an intern's current tasks, skills, and goals. However, condensing all these external tools into a centralized and easy-to-use database, will in turn aid to streamline the internship management process.

Per the purpose of any internship program, its aim is to create a pipeline of interns who will eventually make it to full-time positions. JB Hunt is a vast company with several application teams and avenues. It would be convenient to have a resource that not only narrows down which interns will be appropriate for a given team but also be used to advertise them.

2.0 Objective

The objective of this project is to create a website to house useful information about the former and current Engineering and Technology interns at JB Hunt. It will in turn assist the members of leadership to make informed decisions in regard to company hires. This convenient application will display the skills that interns have gained throughout their JB Hunt journey in order to complement a smooth transition to a full-time team. The application will also help organize the interns in a way that is beneficial to full-time employees.

3.0 Background

3.1 Key Concepts

Azure DevOps is a suite of tools that support the software development process. It includes requirements management, project management, testing features, and automated integration/deployment. However, its primary use will be version control. Dev Ops offers git-based version control that allows developers to make their own branch and push changes to the main codebase. It also provides means of reverting the code to any previous state if needed. Subscriptions are free for up to 5 users.

Visual Studio Code is a popular IDE with debugging, auto-code completion, and git functionality. A variety of plugins are available that complement or enhance its features.

Angular is a web application framework that uses typescript, HTML, and CSS. A typical angular app is built with custom “components.” These components are comprised of a template(HTML), class(typescript/), and styles(CSS). Rather than having several web pages per a usual website, an angular application builds upon the main component. The user essentially views the same web page the entire time. The changes the user sees correspond to the swapping and manipulation of individual components.

MongoDB is a document database with scalability and flexibility which is essential to the querying process. It is an open-source NoSQL database management program. It provides drivers for 10+ languages. It stores data in flexible, JSON-like documents, which means fields can vary from document to document and data structure can be changed over time.

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on a JavaScript Engine and executes JavaScript code outside a web browser, which was designed to build scalable network applications.

3.2 Related Work

As mentioned above, this application will serve a similar purpose as existing websites such as LinkedIn, minus the social media component. Frankly, the concept of a platform used to house a large number of individual resumes is not an original idea. However, this application will become a proprietary asset to JB Hunt Transport Services. It could be modified freely to meet company needs. If proven to be successful, expansions/customizations could append any functionalities mainstream resume websites currently possess.

4.0 Design

4.1 Requirements/Design Goals

- Register/Login

InternApp [Home](#) [Create Account](#) [Login](#)

username

email


password

user_roles [for demo purposes]

[create account](#)

- **Intern Profile**

- Profile Picture
- Team/Team History
- Availability
- Hometown
- Hobbies
- Resume

 Benjamin Thiele Team: Eagle Status: Active	Additional Information
	Start Date: 12-13-2022 End Date: 12-15-2022 School: University of Arkansas Previous Team(s): Match, CICD Hometown: Bella Vista, AR Hobbies: Music, Shooting

Ben Thiele
35 Tibberton Circle Drive | Bella Vista, AR 72714 | 337-534-3428 | bthiele@uark.edu

SKILLS SUMMARY

- Programming languages: C#, C++, Java, JavaScript, Python, SQL

RELEVANT COURSEWORK

Programming Foundations I	Operating Systems	Software Engineering
Programming Foundations II	Digital Design	Database Management
Programming Paradigms	Computer Organization	Algorithms

EDUCATION

Bachelor of Science in Computer Science University of Arkansas, Fayetteville, AR GPA 3.36 Major GPA 3.765	Expected: December 2022
Master of Music in Classical Guitar Performance University of Louisiana at Lafayette, Lafayette, LA GPA 3.8	Graduated: December 2014
Bachelor of Arts in Classical Guitar Performance University of Arkansas at Little Rock, Little Rock, AR GPA 3.76	Graduated: May 2012

Main Profile View

- **Search Feature**

- Filter interns with a live search bar with pagination (matches results from intern attributes as the user types)

The screenshot shows a search interface for interns. At the top, there is a search bar labeled "Search Interns". Below it, a section titled "Interns List" contains a table with five rows of names: Benjamin Thiele, Cayla Johnson, Will Jackson, Abby Sharp, and Zedd Williams. The first row is highlighted in blue. Below the table is a pagination control with "« Prev", a blue box containing "1", "2", and "Next »". A red button labeled "Remove All" is positioned below the pagination. To the right of the list is a profile card for Benjamin Thiele, featuring a circular profile picture, the name "Benjamin Thiele", "Team: Eagle", "Status: Active", and a yellow button labeled "Go to Benjamin's Page".

List view with live search. Users can select a user from the list and preview their information.

- **Add Intern Feature**

- Form which allows the addition of an intern into a database.

The screenshot shows the "New Intern" form in a web application. The header includes "InternApp" and navigation links "Home" and "Update Profile". The user's profile "bthiele's Profile" and role "ROLE: user" are visible, along with a "Logout" link. The form is titled "New Intern" and is divided into two sections: "1 Basic Info" and "2 Team Info". The "Basic Info" section contains input fields for "Benjamin", "Thiele", "School *", "Hometown:" (with "Bella Vista" entered), and "AR". The "Team Info" section is partially visible at the bottom.

- **Edit Intern Feature**

- Form which allows the modification of an intern into a database.

The image shows a web form titled "Edit Profile" with two main sections. The first section, "1 Basic Info", contains three text input fields: "Benjamin", "Thiele", and "University of Arkansas". Below these is a "Hometown:" label followed by two more text input fields: "Bella Vista" and "AR". The second section, "2 Team Info", contains one text input field: "Eagle". Below this is a "Team History:" label, followed by two buttons: a green "+ Add Team" button and a red "Reset" button. Below the buttons is a "Team 1:" label and one text input field: "Match".

4.2 Detailed Architecture

The goal is to implement a web application that aids in the management of interns at the internship program at JB Hunt. In a way, sort of like an in-house LinkedIn but without the social media aspect. For our design, we decided to start off the website with a Login page for the interns. From there, a profile page will appear for the logged-in intern. This page will display

basic information like the intern's profile picture, availability, current team, hometown, hobbies, and resume. The Intern list page will include a search bar with filters so that full-time employees can easily find an intern and display their current status or the current task they are working on.

Now, to accomplish all this, we decided to use Angular as the front, and then have Node.js as our backend. With Node.js, we are able to use one of the most popular web frameworks called Express which supports routing, middleware, and view systems for example. We also plan to use Sequelize as a promise-based Node.js Object-Relational Mapping (ORM). We were unable to successfully implement admin authority for the application. The admin authority would allow them to add and delete intern accounts. In addition to admin authentication, we would have liked to successfully implement password recovery and toast messages/notifications.

While completing this project, we ran into quite a few roadblocks. One of the biggest challenges was organizing how we wanted to tackle the project. We spent a lot of time working on understanding the applications that we were using and did not spend a lot of time on how to effectively split up the work. This was exceptionally challenging due to us having to move deadlines around in order to make sure we had a great groundwork going before we started on the actual implementation. Our goal was to have the application up and running by the beginning of December but we did not meet that goal. Going forward, we are aware of how important it is to establish a framework when implementing software. On the more technical side, we also learned the process of storing an image and pdf in a database. For our database, we used MongoDB. Storing an image in the database proved to be a big challenge. Our takeaway is that it seems to be impractical for web apps. They take up a lot of space and could pose threats to the information in the database. We really wanted to include profile pictures as we felt that it was an important feature for our intern app. There were a lot of workarounds that had to be done in order to ensure an effective image storage component. We learned how to implement dynamic forms and pagination which were new concepts as we have not had to implement such features previously. For example, we were able to have a certain number of interns into different pages to navigate through, instead of having hundreds of interns in one page and navigating through that page by scrolling.

Front End Architecture Description

The frontend consists of four primary object types: components, helpers, services, and models. Models are where data structures are defined, and they often mirror the data being sent by the backend that you are communicating with. For instance, *intern.model* is the client side model that represents an Intern, and this mirrors the server side model of an Intern that is sent by the database upon request.


```
app / components / main-card / main-card.component.ts main-card.component.ts main-card.component.ts
Will, 2 hours ago | 2 authors (Benjamin Thiele and others)
1 import { Component, Input, OnInit } from '@angular/core';
2 import { Intern } from 'src/app/models/intern.model';
3 import { InternService } from 'src/app/services/intern.service';
4 import { ActivatedRoute, Router } from '@angular/router';
5
6 @Component({
7   selector: 'app-main-card',
8   templateUrl: './main-card.component.html',
9   styleUrls: ['./main-card.component.css']
10 })
11
12 export class MainCardComponent implements OnInit {
13
14   @Input() viewMode = false;
15
16   @Input() imageToShow: any = null; Will, 2 hours ago • 12_14_2023
17   @Input() currentIntern: Intern = {
18     _id: '',
19     firstname: '',
20     lastname: '',
21     team: '',
22     dateJoined: new Date(),
23     dateLeaving: new Date(),
24     internAvailable: false,
25     city: '',
26     state: '',
27     status: '',
28     previousTeams: [],
29     interests: [],
30     username: ''
31   };
32
33   message = '';
34
```

Component Logic/Typescript

```
68
69 <div class="row">
70   <div class="column">
71     <div class="card">
72       <app-main-card [imageToShow]="imageToShow"></app-main-card>
73     </div>
74   </div>
75
76   <div class="column">
77     <div class="card">
78       <app-additional-card></app-additional-card>
79     </div>
80   </div>
81 </div>
82 <div class="row">
83   <div class="column">
84     <div class="card">
85       <pdf-viewer [src]="pdfToShow"
86         [render-text]="true"
87         [original-size]="false"
88         [show-borders]="true"
89         style="width: 775px; height: 500px">
90     </pdf-viewer>
91   </div>
92 </div>
93 </div>
94
```

Component HTML - Data passed into child components.

```

1 export class Intern {
2   _id?: any;
3   firstname?: string;
4   lastname?: string;
5   team?: string;
6   dateJoined?: Date;
7   dateLeaving?: Date;
8   internAvailable?: boolean;
9   school?: string;
10  city?: string;
11  state?: string;
12  status?: string;
13  previousTeams?: string[];
14  interests?: (string | null)[];
15  username?: string;
16 }
17

```

Intern Model

The backbone of the app is the JWT token. This was the tool that allowed us to implement authorization and authentication for requests made to the backend. It authenticates by taking the username and password packaged in the header of a request and querying the database to ensure that they matched. Similarly, authorization was implemented by reading the 'roles' key-value pair that was packaged in the header of every HTTP request sent from the front end. If a user had 'role' moderator, any request was valid, but if they had only role 'user', all routes except those related to the user adding and updating their own information were disallowed. The file *http.interceptor* is what adds the JWT token to the header of outbound requests.

```

1 import { Injectable } from '@angular/core';
2 import { HttpEvent, HttpInterceptor, HttpHandler, HttpRequest, HTTP_INTERCEPTORS } from '@angular/common/http';
3 import { Observable } from 'rxjs';
4 import { StorageService } from '../services/token-storage.service';
5
6 const TOKEN_HEADER = 'x-access-token';
7
8 @Injectable()
9 export class HttpRequestInterceptor implements HttpInterceptor {
10  constructor(private browserToken: StorageService) {}
11
12  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
13
14    let req1 = req;
15    const token = this.browserToken.getToken();
16
17    if (token != null) {
18      req1 = req.clone({
19        headers: req.headers.set(TOKEN_HEADER, token),
20      });
21    }
22
23    return next.handle(req1);
24  }
25 }
26
27
28
29 export const httpInterceptor = [
30   { provide: HTTP_INTERCEPTORS, useClass: HttpRequestInterceptor, multi: true },
31 ];

```

Helper - HTTP interceptor that attaches browser information to data sent to the database.

In Angular, services are the means by which Angular ‘pages’ gain access to data. Any communication from a page to a model or the backend is handled by a service. The services utilized by our application are: *auth.service*, *image.service*, *intern.service*, *token-storage.service*, and *upload.service*.

```
Will, 3 days ago | 2 authors (Benjamin Thiele and others)
1 import { Injectable } from '@angular/core';
2 import { HttpClient } from '@angular/common/http';
3 import { Observable } from 'rxjs';
4 import { Intern } from '../models/intern.model';
5
6 const baseUrl = 'http://localhost:8080/api/interns';
7
Will, 3 days ago | 2 authors (Benjamin Thiele and others)
8 @Injectable({
9   providedIn: 'root'
10 })
11 export class InternService {
12
13   constructor(private http: HttpClient) { }
14
15   getAll(): Observable<Intern[]> { Benjamin Thiele, 2 months ago * Front End
16     return this.http.get<Intern[]>(baseUrl);
17   }
18
19   get(id: any): Observable<Intern> {
20     return this.http.get(`${baseUrl}/${id}`);
21   }
22
23   findByUsername(username: any): Observable<Intern> {
24     console.log("making request to " + baseUrl + "/user/" + username);
25     return this.http.get(`${baseUrl}/user/${username}`);
26   }
27
28   // username-resume or username-pfp
29   getPic(indentString: any): Observable<Intern> {
30     return this.http.get(`http://localhost:8080/files/${indentString}`);
31   }
32
33   create(data: any): Observable<any> {
34     return this.http.post(baseUrl, data);
35   }
36 }
```

Data Service Example - Intern Service that makes API calls pertaining to intern records.

The upload service takes a file of any size and splits it into multiple HTTP POSTs that are sent to the backend.

The auth service handles outbound HTTP requests for logging in, logging out, and user registration. It also handles the associatedIntern flag, which determines whether a user has access to the create-intern route or the update-intern route.

The token-storage service allows parts of the application to access information about the user that is currently stored in browserStorage. It is the means by which the current user’s state is saved (ie. the user staying logged in if they haven’t chosen to log out).

The image service accesses the endpoint where a user’s images are stored.

The intern service handles HTTP requests from Angular that are meant to send data to the Interns table in the database.

In Angular, components are groups that are composed of html, css, and TypeScript files. The css determines the styles of elements displayed by the html files, and TypeScript files handle performing functions called in the html or populating data that the html needs to render. Below is the *main-card* component.

In addition, *app.html* is what initializes the app when its url is initially navigated to, and it ensures that every child component has the imports and libraries it needs to function.

Two import libraries used were ngx-pagination and n2-search-filter. We implemented client side pagination of the Intern list using ngx, and the search bar that filters the displayed interns was handled by n2-search-filter.

Back End Architecture Description

The backend, built using MongoDB, Node.js, and Express, has four primary components: the router, middlewares, controllers, and models. The router reads incoming HTTP requests and routes to the appropriate controller, or to a middleware if it is defined in the route.

```
router.post(
  '/',
  [
    authJwt.verifyToken,
    authJwt.isUserOrMod,
    validateIntern.checkDuplicateIntern,
  ],
  interns.create,
);
router.post('/update', [authJwt.verifyToken, authJwt.isUserOrMod]);
router.get(
  '/',
  [authJwt.verifyToken, authJwt.isModerator],
  interns.findAllByLastName,
);
router.get(
  '/firstNameSearch',
  [authJwt.verifyToken, authJwt.isModerator],
  interns.findAllByFirstName,
);
router.get(
  '/teamSearch',
  [authJwt.verifyToken, authJwt.isModerator],
  interns.findAllByTeam,
);
```

Middlewares are how access to the backend is moderated. If a path is defined as a ‘moderator-only’ path, then a middleware function is called before granting access to the controller that ensures the request has the moderator role stored in its header. One primary middleware is the upload middleware, which takes a request to upload and formats the new name of the file such that it represents the user who uploaded it (ie. user “will”’s profile picture upload would be stored in the database as will-pfp.jpg) . It also reads from a config file to determine which bucket to upload to, depending on the file type of the request.

```
if (
  match.indexOf(file.mimetype) === 0 ||
  match.indexOf(file.mimetype) === 1
) {
  return {
    bucketName: dbConfig.imgBucket,
    filename: `${request.username}-pfp`,
  };
}

if (match.indexOf(file.mimetype) === 2) {
  return {
    bucketName: dbConfig.pdfBucket,
    filename: `${request.username}-resume`,
  };
}
```

The *authJwt* middleware is the primary file that handles authorization and authentication. This middleware reads the JWT token that is attached to each incoming request, and then determines if a user is allowed to access content depending on the key-values pairs within said token. Below is the *verifyToken* method of the *authJwt* class, which parses tokens that are attached to incoming requests, and rejects requests that lack a token. It also checks this token against a secret to ensure it is valid.

```

verifyToken = (request, res, next) => {
  // Let token = req.session.token;
  const token = request.headers['x-access-token'];
  console.log('verify token');
  if (!token) {
    // Console.log("!token apparently");
    return res.status(403).send({message: 'no token received by backend'});
  }

  jwt.verify(token, config.secret, (error, decoded) => {
    if (error) {
      return res.status(401).send({message: "token doesn't match secret"});
    }

    request.userId = decoded.id; // Makes attributes available to upload_middleware
    request.username = decoded.username;
    console.log('token verified');
    next();
  });
});
};

```

Controllers are devices that make changes to information stored in the database. For example, the “Intern controller” utilizes Mongoose to modify data in the Interns table, and the type of modification is dependent on the method that the router routes the request to. Below are a couple methods from that controller. These are effectively the same methods, just with different ways of accessing the ‘current user’.

```

// Returns '1' user by username
exports.findByUsername = (request, res) => {
  const searchUser = request.query.username;
  Intern.find({username: searchUser})
    .then((data) => {
      res.send(data);
    })
    .catch((error) => {
      res.status(500).send({
        message: error.message || 'intern.intern.findByUsername error',
      });
    });
});

exports.findByUsernameParams = (request, res) => {
  const searchUser = request.params.username;
  Intern.findOne({username: searchUser})
    .then((data) => {
      res.send(data);
    })
    .catch((error) => {
      res.status(500).send({
        message: error.message || 'intern.intern.findByUsername error',
      });
    });
});
};

```

Models define the schema of data in the database. It is what formats tables. Below is the User schema. Every model imports the 'mongoose' package, because this is how Express interfaces with MongoDB services. The datatypes and data structure are described here using json.

```
const mongoose = require('mongoose');

const User = mongoose.model(
  'User',
  new mongoose.Schema({
    username: String,
    email: String,
    password: String,
    associatedIntern: Boolean,
    roles: [
      {
        type: mongoose.Schema.Types.ObjectId,
        ref: 'Role',
      },
    ],
  })
);

module.exports = User;
```

4.3 Risks

Risk	Risk Reduction
Authorization - occurs after confirming the identity of the user, the task is then to find out which services they have access to	We mitigate this risk by giving roles to the interns and the admins/employees in the company. For example, for interns, they would have a user role which will allow them to see their own profile page with their information only. While the admins/employees will have a moderator role and be able to see the list of each intern and see their status or what task they are working on or see their profile page.

4.4 Tasks

1. Meet with the Project lead at JB Hunt - Discuss what is expected of the website and our plans for how it should look
2. Research - Understand & gain knowledge on the applications being used to complete the project (i.e. find any helpful tutorials about Angular & MongoDB)
3. Create Design Document - Outline how the application should operate using lucid chart. Figure out how the applications will work/communicate with each other in order to make a cohesive web application
4. Implementation of website
 - a. Implementation of backend functionalities so that an intern can be successfully added to the database
 - b. Implementation of the front end to show an intern being added and then populating an “Intern List” to keep track of all the interns that have been added
 - c. Implementation of front-end profile page so that a user can access a profile page from the intern list.
 - d. Populate the profile page with angular cards to display basic information about the intern (i.e. name, profile picture, the date the user joined, current team/team history, hometown, interests, and resume)
 - e. Edit the profile page so that it looks more “clean” and cohesive with the rest of the application
 - f. Edit the add intern page so that the intern can edit and add their details to their profile page
 - g. Implementation of pagination for the intern list so that the user can choose how many interns to see at a time without having to scroll all the way down on just one-page
 - h. Implementation of successfully allowing users to upload images for the profile picture and PDFs for the resume
5. Test website - Make sure all functionalities of the website are up to par by creating test profiles to verify usability
6. Document results - Report what went well and gather feedback via survey from users after using the application

4.5 Schedule

Tasks	Dates
1. Research/Organizing - understand applications being used, assign tasks to each member	8/30-9/7
2. Design website - Each member outlines their specific task and then as a group comes together to decide what works best	9/7-9/21
3. Implementation of website - Each member completes their portion of the website (front-end/back-end components). Azure DevOps will be utilized to ensure version control. Weekly meetings will be held to discuss progress thus far(in person or via teams)	9/21-11/13
4. Test website - Make sure all functionalities of the website are up to par via creating test profiles to verify usability	11/13-12/11
5. Document results - Report what went well and gather feedback	12/11-12/14

4.6 Deliverables

- Design Document: Contains a listing of each major hardware and software component and layout for application
- Database side, MongoDB
- Node.js for the Backend
- Angular for the Frontend
- Final Report

5.0 Key Personnel

Students

Cayla Johnson is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. She has completed relevant courses. She has completed two out-of-state internships with Google and HSBC with varying roles in the technological and business space. In regard to this project, she is responsible for the implementation of the front-end implementation of the website.

Ben Thiele is a senior Computer Science major in the CSCE department at the University of Arkansas. He has completed a variety of relevant courses such as Database Management and Software Engineering. He also has experience as a TA for Programming Paradigms and Programming Foundations II. Ben is currently an annual intern at J.B Hunt Transport Services, Inc. where he plans to begin his career as a software engineer. Over the course of this project, he will primarily focus on front-end web development and the general implementation strategy.

Jessi Soto is a senior Computer Engineering major in the CSCE department at the University of Arkansas. He has completed a wide range of relevant courses such as System Synthesis and Modeling, Embedded Systems, and Computer Architecture. As for the project, he will be responsible for the front end and some backend implementation of the website.

William Jackson is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He is competent in several programming languages and paradigms, particularly in the realm of game development. He was the head backend designer for his capstone project.

Industry champion

Reese Stanley is a director of Engineering and Technology at JB Hunt. He currently leads the application development internship program. He attended the University of Arkansas where he

received a B.A. in business management and a minor in information technology. A long-time JB Hunt employee, with origins in software development.

5.0 Facilities and Equipment

The majority of this project will be accomplished remotely.

7.0 References