

Team Blue

Career Tracker

Software Detailed Design

William Ntumba, Scott Shrout, Anthony Freitas

Table of Contents

1. Data Design	2
1.1 Overview	2
1.2 User Interface	2
1.2.1 Data Objects	2
1.2.1.1 UserAuthentication	3
1.2.1.2 AlumniRecords	3-6
1.2.1.3 EmployerRecords	6-7
1.2.1.4 GraduateSchoolRecords	7
1.3 Service Side Services	8
1.4 Data Store	8
1.4.1 Database Schema	8-9
2. Architecture Design	10
2.1 Overview	10
2.2 Architectural Models	11
2.2.1 Architecture Block Diagram	11
2.2.2 Data Flow Diagram	12-13
2.2.3 Data Mapping Diagram – Alumni Import File	14
2.2.4 Hardware-Software Mapping – Deployment Diagram	15-16
3. Interface Design	17
3.1 Overview	17
3.2 User Interface Process Flow	17
3.3 User Authentication Page	18
3.3.1 User Authentication Page Description	18
3.4 Account Creation Page	19
3.4.1 Account Creation Page Description	19
3.5 Alumni Records Page	20
3.5.1 Alumni Records Page Description	20
4. Procedural Design	21
4.1 Overview	21
4.2 Operations & Procedures	21
Individual Contributions	22
Key Personnel Information	22

1. Data Design

1.1 Overview

The Career Tracker application consists of three main subsystems:

- The user interface, allowing a user to interact with the application using web browser software
- Server-side services, allowing for processing of CSV and report files
- Data store, a relational database in which the data used in the application is stored and retrieved

1.2 User Interface

The user interface is a web application written in Angular, HTML, CSS, Typescript, and Javascript.

As with most Angular applications, this application will be designed using components. The components represent identifiable parts of the overall application, each implementing particular functionality.

In Angular, components generally contain:

- The application logic, written in Typescript
- CSS layout files
- View file, largely composed of HTML

In order to facilitate the transfer of application data from the data store to the view, Angular allows for property binding using Javascript data objects. These may be single objects or multiple objects contained in array collections.

1.2.1 Data Objects

Data objects used in the user interface are described within the context of the component in which they are used. Data objects are shown in Javascript Object Notation (JSON).

1.2.1.1 UserAuthentication

The UserAuthentication component is responsible for passing the submitted username and password credentials to the server for validation. The password entered on the form will be hashed and the hashed value compared to the existing value in the database. Thus, this component only involves the transfer of the below data elements:

```
{  
    user_id: a unique string value assigned for a user of the system,  
    password: a string representation of the user's hashed password,  
}
```

1.2.1.2 AlumniRecords

The AlumniRecords component allows a user to search for, view, and edit alumni records containing address, telephone, e-mail, job/employer, and post-graduate school information. The data design is as shown below.

```
{  
    student_id: a unique, system generated value used to identify a specific student record,  
    first_name: the first name of a student,  
    last_name: the last name of a student,  
    phone_num: the current primary phone number associated with the student,  
    email: the current primary non-university e-mail associated with the student,  
    address: {  
        address_line_1: the first line of the student's mailing address,  
        address_line_2: the second line of the student's mailing address,  
        city: the city in which the student's mailing address is located,  
        state: the state in which the student's mailing address is located,  
        zip_code: the zip code associated with the student's mailing address,  
    },  
    added_by: the id of the user that added the record,  
    added_date_time: the date and time on/at which the record was added,
```

updated_by: the id of the user that last modified the record,

updated_date_time: the date and time on/at which the record was last modified,

employments:

```
[  
  {  
    employer: {  
      employer_name: the name of the employer,  
      address: {  
        address_line_1: the first line of the employer's address,  
        address_line_2: the second line of the employer's address,  
        city: the city in which the employer is located,  
        state: the state in which the employer is located,,  
        zip_code: the zip code associated with the employer,  
      },  
      job_title the job title associated with the employment,  
      active: an indicator of whether the employment is current or past,  
      added_by: the id of the user that added the record,  
      added_date_time: the date and time on/at which the record was added,  
      updated_by: the id of the user that last modified the record,  
      updated_date_time: the date and time on/at which the record was last modified  
    }  
  ],
```

degrees:

```
[  
  {
```

```

    diploma_description: a description of the attained degree,
    graduation_term_code: the graduation term code associated with the degree
    (from import file),
    added_by: the id of the user that added the record,
    added_date_time: the date and time on/at which the record was added,
    updated_by: the id of the user that last modified the record,
    updated_date_time: the date and time on/at which the record was last modified
    }
  ],
  graduate_schools:
  [
    {
      school_name: the name of the graduate school associated with
      the student,
      city: the city in which the graduate school is located,
      state: the state in which the graduate school is located
    }
  ],
  comments: [
    {
      comment: the comment entered by the user,
      added_by: the id of the user that added the record,
      added_date_time: the date and time on/at which the record was
      added
    }
  ]
}

```

1.2.1.3 EmployerRecords

The EmployerRecords component allows maintenance of basic information related to the employers that are associated with students. The data layout is shown below.

```
{  
  
  employer_id: a unique identifier associated with an employer record  
  
  employer_name: the name of the employer  
  
  address: {  
  
    address_line_1: the first line of the employer's address,  
    address_line_2: the second line of the employer's address,  
    city: the city in which the employer is located,  
    state: the state in which the employer is located,  
    county: the county in which the employer is located,  
    zip_code: the zip code associated with the employer  
  
  },  
  
  added_by: the id of the user that added the record,  
  
  added_date_time: the date and time on/at which the record was added,  
  
  updated_by: the id of the user that last modified the record,  
  
  updated_date_time: the date and time on/at which the record was last modified,  
  
  comments: [  
    {  
  
      comment: the comment entered by the user,  
  
      added_by: the id of the user that added the record,  
  
      added_date_time: the date and time on/at which the record was  
      added  
  
    }  
  ]  
}
```

```
}
```

1.2.1.4 GraduateSchoolRecords

The GraduateSchool component allows maintenance of basic information related to the graduate schools that are associated with students. The data layout is shown below.

```
{
```

graduate_school_id: a unique identified associated with the graduate school record,

school_name: the name of the graduate school,

city: the city in which the graduate school is located,

state: the state in which the graduate school is located,

added_by: the id of the user that added the record,

added_date_time: the date and time on/at which the record was added,

updated_by: the id of the user that last modified the record,

updated_date_time: the date and time on/at which the record was last modified,

comments: [

```
{
```

comment: the comment entered by the user,

added_by: the id of the user that added the record,

added_date_time: the date and time on/at which the record was added

```
}
```

```
]
```

```
}
```


1.3 Server-Side Services

The server-side services subsystem is responsible for:

- Processing CSV input files submitted by a user through the user interface. The files, which contain details related to alumni that are provided to the Office of the Dean by the Office of the Registrar, will be parsed and applicable updates processed to the backend database.
- Generation of on-demand reporting and output files for user consumption through the user interface

The server side services will use the data objects as shown in the preceding **AlumniRecords**, **EmployerRecords**, and **GraduateSchools** component subsections.

1.4 Data Store

The data store used in this application consists of a relational database management system (RDBMS) used to store and retrieve application data.

1.4.1 Database Schema

The database schema is shown in the following database relationship diagram.



2. Architecture Design

2.1 Overview

The Career Tracker application is a web application that allows a user to search for, view, and edit data assets stored in a backend database using any modern web browser client on a laptop or workstation computing device. In addition to the user interface, some server-side processing is used to extract data from imported data files as well as generate reporting and output files.

The main physical components associated with the application include:

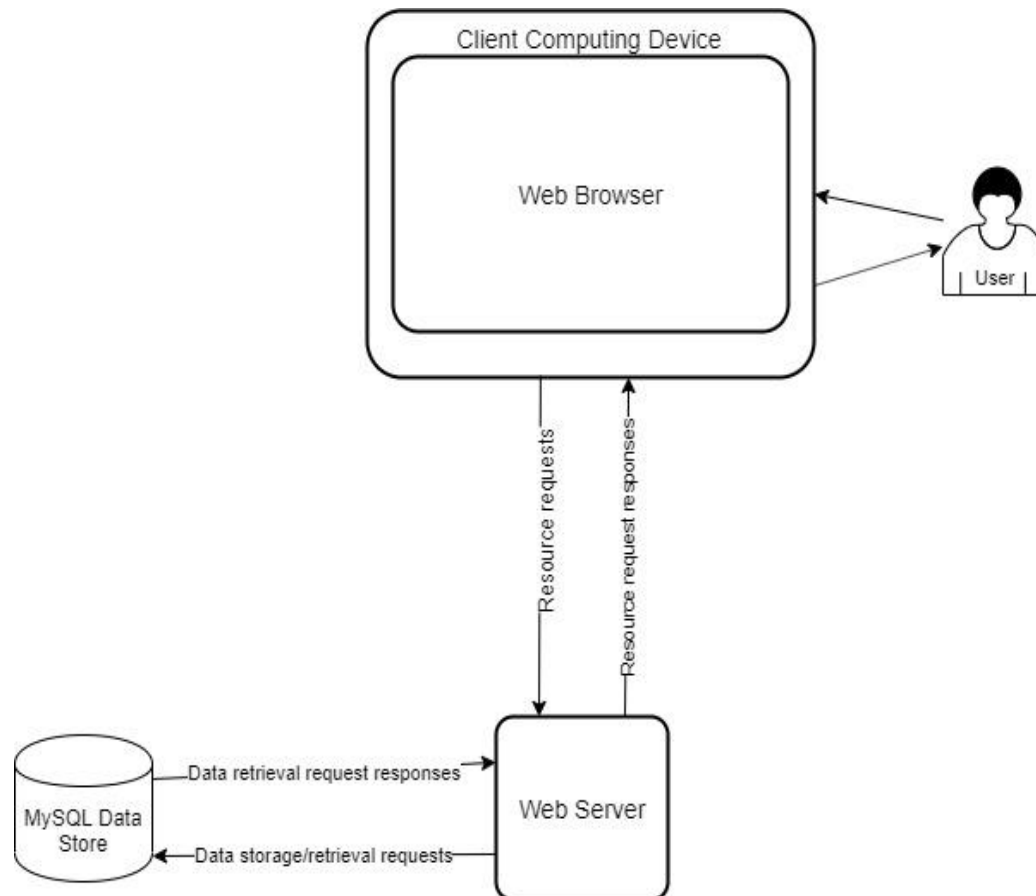
- **Application/Web Server** – the server at which resource requests are received and processed
- **Database Server** – the server that stores and retrieves data used in the application
- **Client Computing Device** – the workstation or laptop on which the application will be accessed by an end user
 - **Web Browser Client** – an application residing on the client computer device used to issue resource requests to the application server and display content back to the end user

Other components required for data transmission, such as network connection devices, transmission lines, and transmission protocols are commonly used in many applications and not detailed extensively here.

2.2 Architectural Models

2.2.1 Architecture Block Diagram

The architecture block diagram shows below shows a high-level view of the application's architecture as well as interactions between components.



2.2.2 Data Flow Diagram

The data flow diagram shown on the following page depicts the major functional components of the application along with the general flow of data among them.

The functional components include:

Search Employer Data – allows a user to search for employer records using field criteria, such as employer name

View Employer Detail – allows a user to view all fields relevant to the employer record

Update Employer Data – allows a user to make updates to the employer record

Search Alumni Data - allows a user to search for alumni records using field criteria, such as student name

View Alumni Detail – allows a user to view all fields relevant to the alumni record

Update Alumni Data – allows a user to make updates to the alumni record

Search Graduate School Data - allows a user to search for graduate school records using field criteria, such as school name

View Graduate School Detail – allows a user to view all fields relevant to the graduate school record

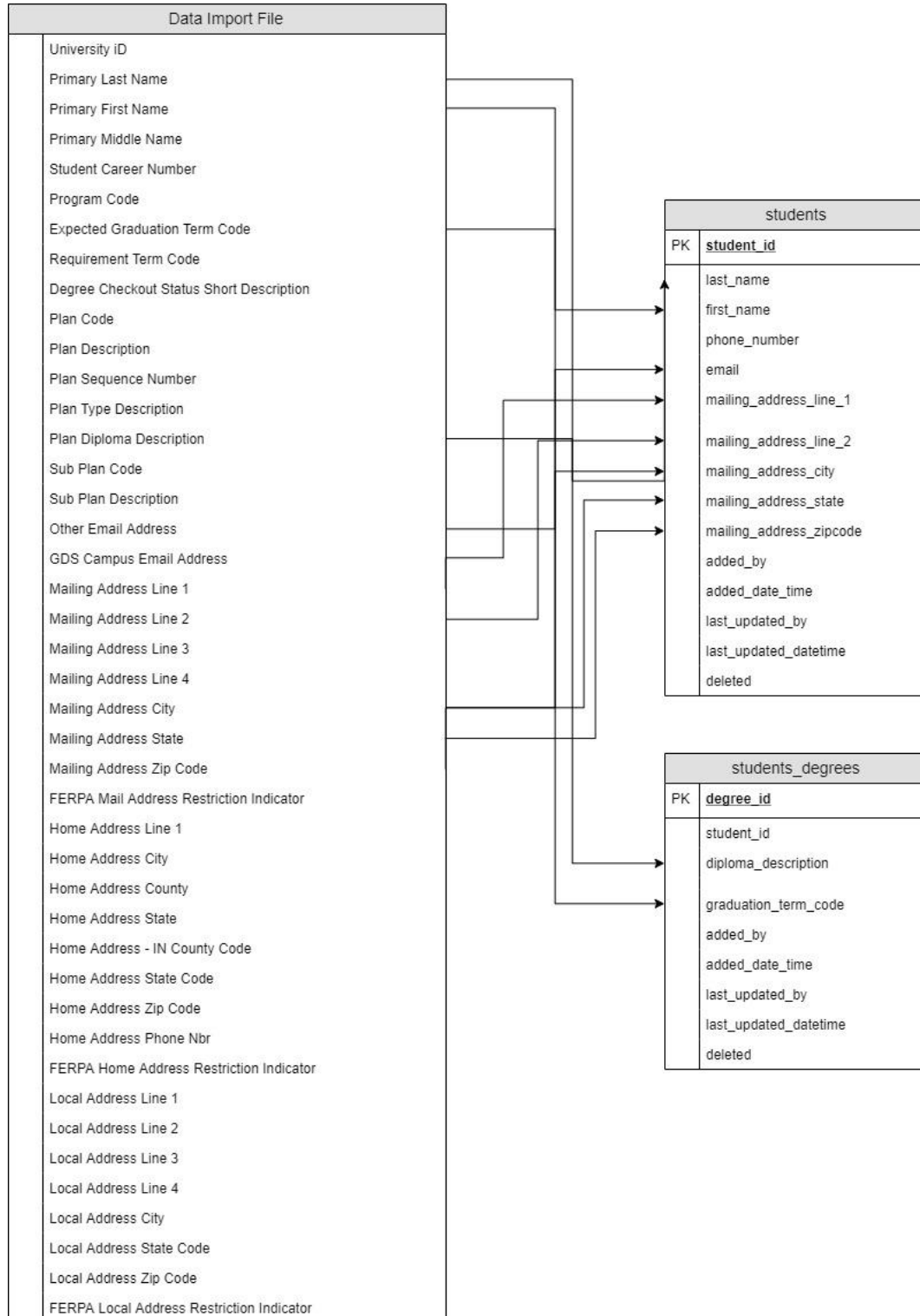
Update Graduate School Data – allows a user to make updates to the graduate school record

File Upload Processing – allows a user to upload a CSV file that will be parsed and updates applied to appropriate data entities

Reports/Export Processing – allows a user to generate reporting and file exports based upon specified criteria

2.2.3 Data Mapping Diagram – Alumni Import File

The diagram below depicts the file-to-database-entity mapping for the alumni import file that will be used in loading data to the application.



2.2.4 Hardware-Software Mapping – Deployment Diagram

The diagram shown on the following page depicts the major physical and logical components of the application.

The components depicted include:

Application Server – The application server is the hardware device on which the web server application runs.

Node JS Server – Node JS Server is an application running on the application server that receives and responds to web client requests.

Server Side Processing – Node JS processes requests for content or file processing requests using pre-defined libraries.

Client Content Request Processing – The web server listens for and responds to requests from clients for web content.

TCP/IP – Transmission control/internet protocol used to establish communication between application and database servers.

Database Server – The database server is the hardware device on which the relational database management system (RDBMS) application runs.

MySQL Server – MySQL Server is a relational database management system used to store and retrieve data using the application.

students – The students table contains general student information, including name, phone number, e-mail address, and address. The student_id is used as a foreign key in several other related tables.

student_employments – The student_employments table contains job entries for a student, including the role name and employer.

employers – The employers table contains the name and address for employers referenced in the Employments table.

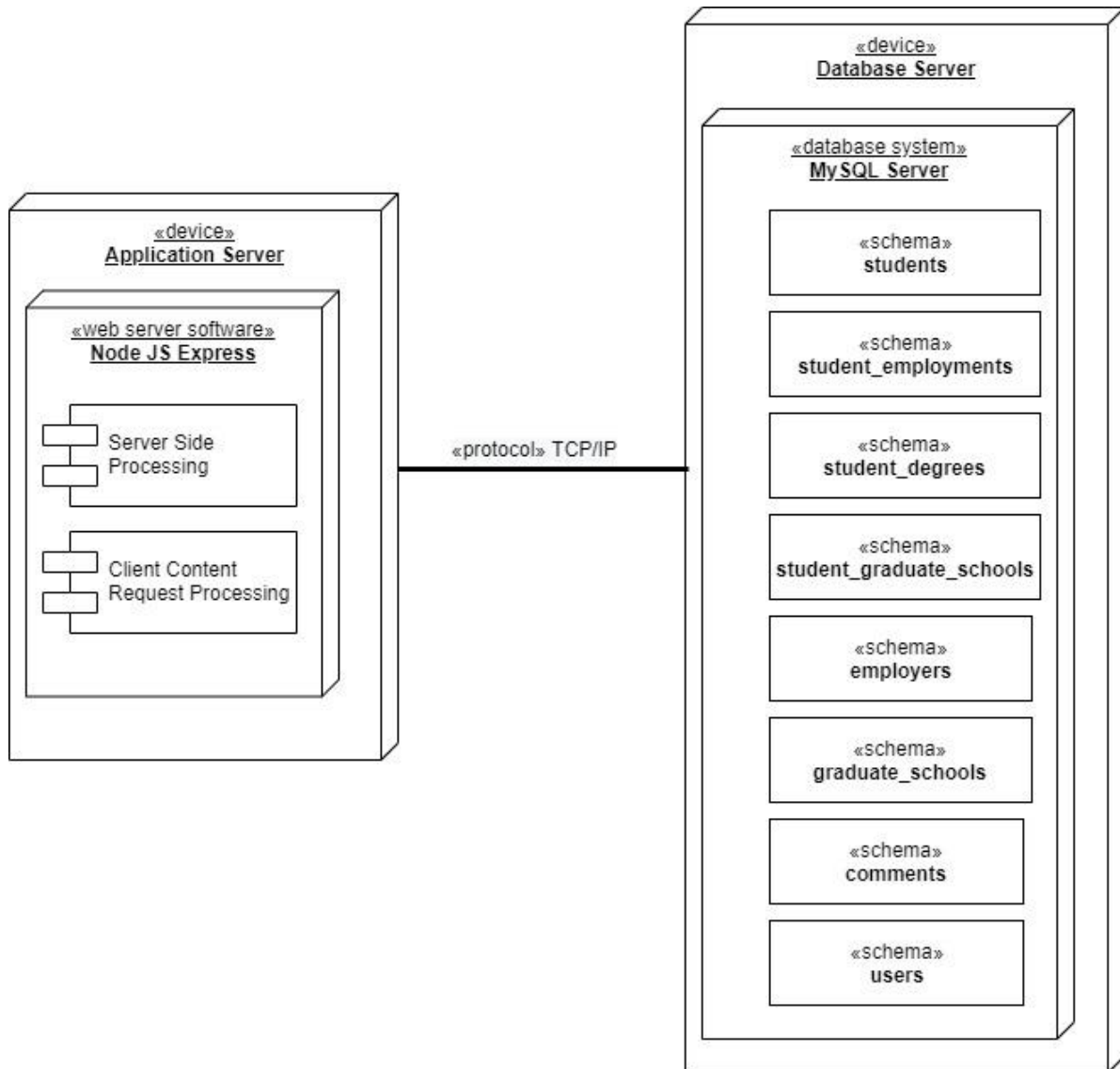
student_graduate_schools – The student_graduate_schools table contains the identifier of graduate schools associated with a student.

graduate_schools – The graduate_schools table contains the name, city, and state of educational institutions providing graduate degree programs.

student_degrees – The Degrees table contains degrees attained by students.

comments – The Comments table contains comments entered by a user related to a student, employer, or graduate school record.

Users – The users table contains information related to users of the application.



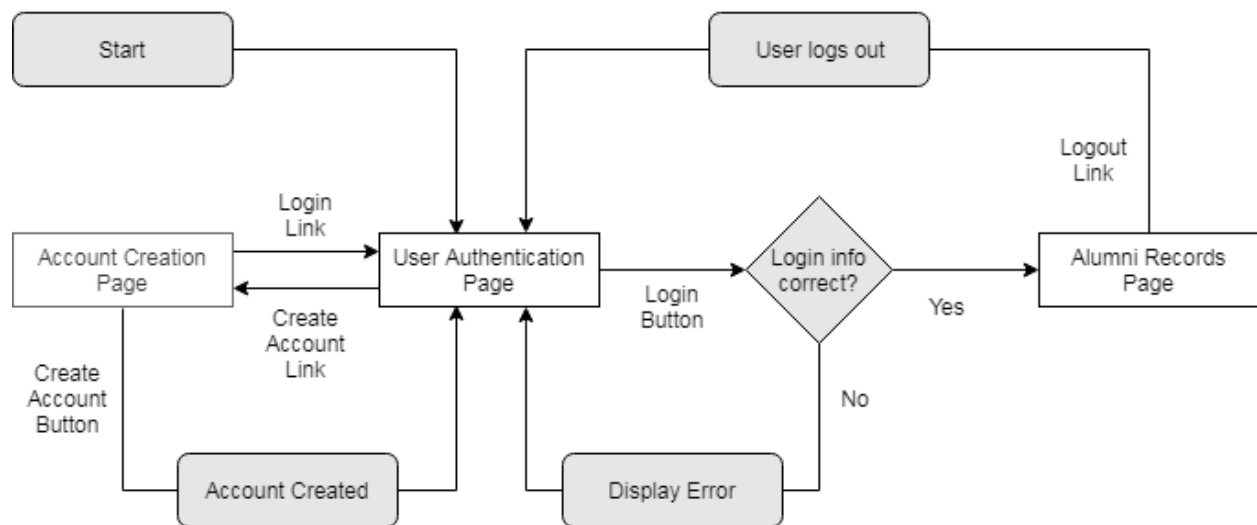
3. Interface Design

3.1 Overview

This portion of the document will cover the development of the user interface for the Career Tracker application. The following sections will give a brief layout of the interface process flow and a few mock application page designs will be displayed to give a graphical representation of what the user will see when interacting with the program.

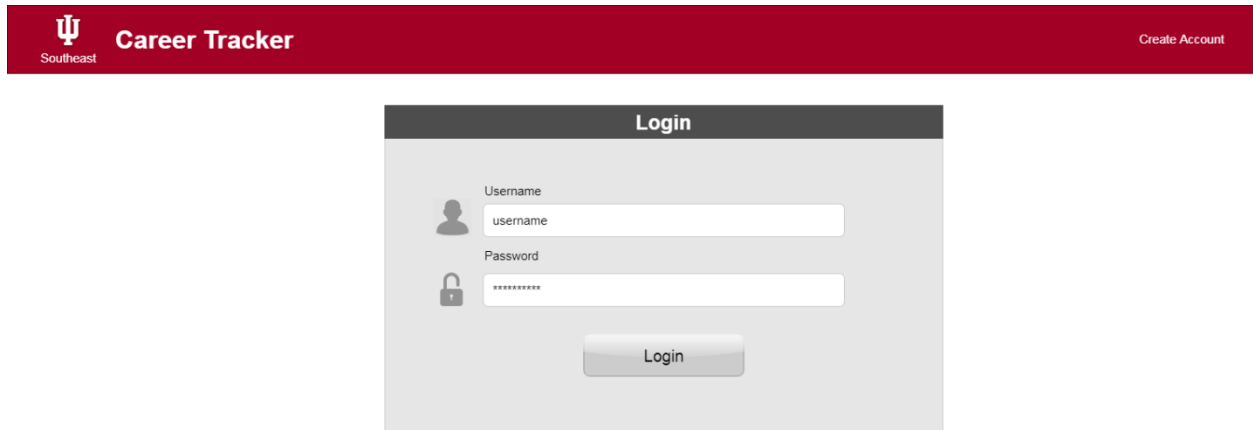
3.2 User Interface Process Flow

The chart below shows the actions the user can take to navigate through the application.



3.3 User Authentication Page

This is a prototype user authentication page and the entry point of the application. The main component is the login form where the user will provide their name and password in order to view the alumni information. This page will likely also contain a link for creating new accounts on the application.



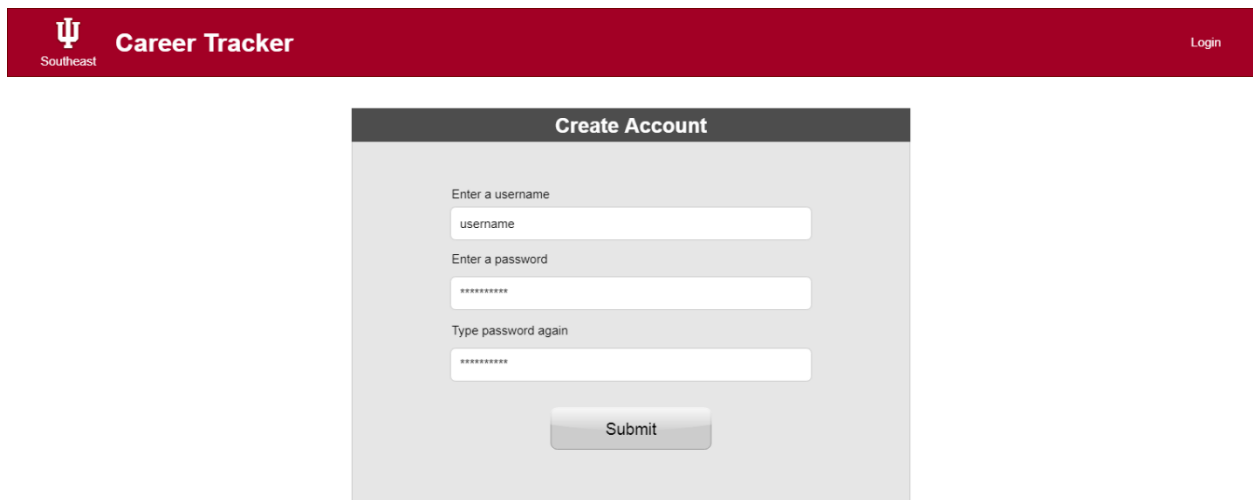
3.3.1 User Authentication Page Description

Purpose: Allow users to login to the application. Only a valid name and password will allow access to the alumni records.

Navigation & User Interaction: The user will type their login information into the appropriate boxes. Once the user has entered their credentials and clicked the login button, they will then be sent to the alumni records page.

3.4 Account Creation Page

This is a sample account creation page. Users will be able to navigate here from the authentication screen. The main component is the create account form where users will create their username and password to login to the application.



The screenshot shows a web application interface for 'Career Tracker'. At the top, there is a dark red header bar. On the left side of the header, there is a logo consisting of a stylized Greek letter Psi (Ψ) above the word 'Southeast'. To the right of the logo, the text 'Career Tracker' is displayed in white. On the far right of the header, the word 'Login' is visible in a smaller white font. Below the header, the main content area is light gray. In the center of this area is a white rectangular box with a dark gray title bar at the top that says 'Create Account'. Inside this box, there are three text input fields. The first is labeled 'Enter a username' and contains the text 'username'. The second is labeled 'Enter a password' and contains a series of asterisks (*****). The third is labeled 'Type password again' and also contains a series of asterisks (*****). Below these three fields is a gray button with the word 'Submit' in white text.


3.4.1 Account Creation Page Description

Purpose: Users will be able add a username and password to the system for login purposes.

Navigation & User Interaction: There will be text boxes available for entering the login information. Users will be asked to enter their password twice to avoid mistypes. After clicking submit, the user's account will then be added to the user database and they will return to the authentication page.

3.5 Alumni Records Page

This is a prototype design for the alumni records interface and the main page of the application. Users will be allowed access to this page after being successfully validated. All relevant information from the database will be displayed on this page in a table.

<div> Career Tracker</div> <div>RecordsImport DataReports</div> <div>Logout</div>							
Database ID	Last Name	First Name	City	State	Graduation Term Code	Current Employer(s)	Graduate School(s)
ID Num	Last Name	First Name	City	State	Graduation Term Code	Current Employer	Graduate School
ID Num	Last Name	First Name	City	State	Graduation Term Code	Current Employer	Graduate School
ID Num	Last Name	First Name	City	State	Graduation Term Code	Current Employer	Graduate School
ID Num	Last Name	First Name	City	State	Graduation Term Code	Current Employer	Graduate School
ID Num	Last Name	First Name	City	State	Graduation Term Code	Current Employer	Graduate School
ID Num	Last Name	First Name	City	State	Graduation Term Code	Current Employer	Graduate School
ID Num	Last Name	First Name	City	State	Graduation Term Code	Current Employer	Graduate School
ID Num	Last Name	First Name	City	State	Graduation Term Code	Current Employer	Graduate School
ID Num	Last Name	First Name	City	State	Graduation Term Code	Current Employer	Graduate School
ID Num	Last Name	First Name	City	State	Graduation Term Code	Current Employer	Graduate School
ID Num	Last Name	First Name	City	State	Graduation Term Code	Current Employer	Graduate School
ID Num	Last Name	First Name	City	State	Graduation Term Code	Current Employer	Graduate School
ID Num	Last Name	First Name	City	State	Graduation Term Code	Current Employer	Graduate School
<div>Search</div> <div><div>Search Category</div><div>▼</div><div>Input Box</div><div>Search</div></div> <div>Previous12345Next</div>							

3.5.1 Alumni Records Page Description

Purpose: This page displays the alumni records and allows users to search and modify the data.

Navigation & User Interaction: A portion of the data from the records is displayed in the table. Users will be able to click on a record to bring up a window containing more detailed information about the selected alumni. The table will be separated into pages that the user can navigate using the previous and next buttons in the lower right corner. A search function is available in the lower left for looking up records with specific criteria. The upper portion of the screen will contain links for importing data, producing reports, and logging out of the application. After logging out, users will then be taken back to the authentication screen.

4. Procedural Design

4.1 Overview

Each procedure begins with a request to the back-end server, Node JS. The server processes it with by making calls to the database. The result or response to the request is then displayed to the screen in a friendly user interface.

4.2 Operations & Procedures

The system allows for the following operations and procedures: import, search, update, and delete. We discuss the use of each in the next lines.

- **Import** – The user can upload the spreadsheet here. Clicking the import button triggers a script to be run to verify the validity of the document. This ensures that the uploaded file is in the correct format and represents the report from the registrar. The script is also responsible for sorting through the spreadsheet and feed the database only with the relevant columns. The database will return the appropriate response to the server whether or not the operation was successful.
- **Edit** – If the user wants to update the database, they would proceed by first searching for the specific record to be amended and perform the necessary modifications. All these operations are requests made to the server that is connected to the back-end database. The modifications are validated with the server to verify that it is correct and in accordance with the school's policies. The user is asked for confirmation when submitting the changes. If the user approves of the changes, then, they are committed to the database. Otherwise, the entire transaction is rolled back, and no data would be modified.
- **Search** – If the user wants to look up a particular alumnus, he or she would provide their information following the search criteria. This procedure queries the database for any matching records.
 - **View details** – this procedure shows more information about a single record at a time. It is invoked after a successful search of a given record.
 - **Delete** – this procedure is responsible for querying the database first, and then performing the necessary deletion of the record or certain fields.
 - **Generate report** – this is the procedure responsible for generating different reports based on the data in the database.

Key Personnel Information

William Ntumba – Development Team, Team Leader

Scott Shrout – Development Team

Anthony Freitas – Development Team

Dana Hope – Project Sponsor

Individual Contributions

Scott Shrout

1. Data Design
2. Architecture Design

Anthony Freitas

3. Interface Design

William Ntumba

4. Procedural Design