3 ResearchLink Specific Requirements
This section details the functional requirements for ResearchLink. It will address the front-end and back-end requirements, to include the algorithms and database specifications.

3.1 Functional Requirements
The functional requirements will specify what is required for the following features: create an account, create and modify profile information, create and modify research opportunities, search for opportunities and user profiles, notification system, and the news highlight newsfeed.

3.1.1 Front-End Functional Requirements

3.1.1.1 - New User Authentication (Matthew Redenius)
This is a web page for a new user to become a registered user. A user name, password, first name, last name, and ODU email will be required for authentication. This information will be saved to the database. When a faculty user creates a profile, their email will be queried in the database against pre populated known faculty emails for authentication. (see section 3.1.2.5 for the back-end requirements for this feature) The following functional requirements must be met:

1. Provide all user types with the ability to input information with the following fields:
   a. First Name
   b. Last Name
   c. Username
      i. No more than 12 characters long
      ii. No special characters
      iii. Cannot already be stored in database
   d. Password
      i. Must be at least 8 characters
      ii. Must contain at least one capital letter
      iii. Must have at least one number
   e. Confirm Password
      i. Must be at least 8 characters
      ii. Must contain at least one capital letter
      iii. Must have at least one number
      iv. Must match the password field
   f. Email (Must have ODU domain)
      i. Cannot already be stored in database
   g. All fields will be blank upon rendering/
h. Provide buttons that allow user to choose between:
   i. Faculty Profile
   ii. Student Profile
i. Provide button for user to submit information for validation.

3.1.1.2 - Manage User Profile Page (Kevin Ashley)

This is a web page form that provides an authenticated user with the capability to edit their profile information based on their designated user type. When the user navigates to this form, the form fields will be pre-populated with data that has been previously saved to the database. The following functional requirements must be met:

1. Provide each user type with the ability to edit personal information with these form fields:
   a. Photo
   b. First name
   c. Last name
   d. Address
   e. City
   f. State
   g. Zip
   h. Email
   i. Phone
2. Provide the Student user type with the ability to edit education information with these form fields:
   a. GPA
   b. Grade Level
   c. Major
   d. Distance Learning
   e. Courses Taken
   f. Interest Areas
3. Provide the Faculty user type with the ability to edit professional information with these form fields:
   a. Department
   b. Courses Taught
   c. Research Affiliations
   d. Interest Areas
4. Provide the ability to save profile changes to database
5. The below algorithm illustrates the logic behind this feature:
3.1.1.3 - Manage Research Opportunity Page (Dinah Watkins)

This is a web page form that provides Administrator and Faculty accounts the capability to create a new research opportunity. In this instance, the user is presented with an empty form. When an authenticated user chooses to edit a research opportunity that they have created, the form fields will be pre-populated with data that has been previously saved to the database. Each record is stored in the database as an active opportunity until the deadline/expiration date has passed. A calendar based feature will assist in the designation of an opportunity being marked as expired and removed from being accessible by active searches. Section 3.1.2.3 contains the back-end requirements for this feature. Database requirements are contained in Section 3.1.2.1.14. The following functional requirements must be met:

1. Provide the capability to create research opportunities to the following user roles:
   a. Faculty user
   b. Administrator
2. Provide the capability to modify information for the following fields:
   a. Title
   b. Description
   c. Categories
   d. Department
   e. Availability for Distance Learning students
   f. Type of opportunity
   g. Paid or unpaid
   h. Payment amount
i. Application start date
d. Application end date
e. Research start date
l. Research end date
m. Expired (Boolean)

3. Provide the capability to store each opportunity entered into the database

3.1.1.4 - Search Opportunities Page (Matthew Redenius)

This is a web page that provides registered users with a display of filtered/ordered research opportunities. Upon navigation to the "Search Opportunities" page, a list of all current opportunities is displayed in order of most recently added. The user will be able to filter out opportunities using keywords as well sort them based on user preference. All research opportunities will be retrieved from the database (see section 3.1.2.6 for the back-end requirements for this feature). The following functional requirements must be met:

1. Provide the capability to populate opportunities using a keyword:
   a. Opportunity fields that will be searched:
      i. Title
      ii. Description
      iii. Category

2. Provide the capability to filter out unpaid opportunities

3. Provide the capability to filter out distance learning opportunities

4. Provide the capability to sort opportunities by:
   a. Date Posted
   b. Date Ending

5. When page is first rendered, opportunities need to be displayed in order of most recently added
3.1.1.5 - User Profile Search Page (Kitt Parker)

This is a web page that provides a registered faculty user the capability to perform profile searches. The web form will contain data search fields that represent specific information associated with user accounts. User accounts only exist for users who have registered within the ResearchLink product. Searches may return zero to any number of results. The web form will allow the faculty user to sort the results by available numerical fields such as user id. The following functional requirements must be met:

1. Allow only registered faculty users
2. Support multiple account returns on successful searches
   a. Provide user with result page pagination of 10 to 100
   b. Provide faculty user with keyword search capability
   c. Interests
   d. Completed courses
   e. Affiliations
3. Provide faculty user with account field search capability
   a. GPA
   b. First name
   c. Last name
   d. Account type
   e. Major
   f. Grade level
   g. City
   h. State
   i. Phone
4. The below algorithm illustrates the logic behind this feature:

![Algorithm Diagram]

3.1.1.6 - Web Interface Notification System (John Raha)

This notification system is a web-based message system which sends notifications to registered users. Notifications include messages related to
research opportunity creation, account status changes, and information related to a user's department. Received messages may be marked as read and deleted by the user. The following functional requirements must be met:

1. The system shall trigger a website notification when there is an email notification.
   a. Trigger a notification to students when a new opportunity has been created matching the student's profile list of interests.
   b. Trigger a notification to department members with new updates.
   c. The system shall allow administrators to send notifications to all registered users.
2. Notifications will contain a pre-written text message summarizing the triggering event.
3. All users will have a message center to access notifications.
4. The notifications will be available in the user's message center.
5. Notifications may be marked as read.
6. Notifications may be deleted by the user.

3.1.1.7 - Email Notification System (John Raha)

This notification system is the email-based message system which sends emails to registered users. All notifications sent to a user's email are the same notifications displayed on the website. Users may turn off email notifications at any time. The following functional requirements must be met:

1. Trigger a notification to students when a new opportunity has been created matching the student's profile list of interests.
2. Trigger a notification to department members with new department news.
3.
4. The system shall allow administrators to send notifications to all registered users. Notifications shall be sent the registered user's ODU email address.
5. Notifications will contain a text message summarizing the triggering event.
6. The below algorithm illustrates the logic behind this feature: o
3.1.8 - News/Highlights Feature (Dinah Watkins)

This is a looping feed that is embedded on a web page within the launch page for ResearchLink and also the landing page upon user login. It displays news/highlight records created by Faculty and Administrator accounts. Each record will be pulled from the database within the allotted time frame they are considered active records. A calendar based feature will assist in archival of news/highlight records that have passed the expiration date (see section 3.1.2.4 for the back-end requirements for this feature). There will be two versions of feeds: public and internal. The following functional requirements must be met for each version:

1. Provide the capability of creating records to the following user roles:
   a. Faculty user
   b. Administrator
2. Provide the capability to submit the record to the database
3. Provide the capability to clear entries made to record being created
4. Provide spell check capability
5. Provide the capability to enter the following information into the fields:
   a. Title
   b. Description
   c. Categories
   d. Department
   e. Available to public (Boolean)
   f. Date Posted
   g. Expiration date
   h. Expired (Boolean)
6. Provide the capability to query the database to retrieve records based on:
   a. Public/internal designation
b. Date posted
7. Provide the capability to request the display of the Public active records:
   a. Departmental
   b. Student news/highlights
8. Provide the capability to request the display of the Internal active records:
   a. Departmental news/highlights
   b. Student news/highlights
   c. New research opportunities
   d. ResearchLink update news

3.1.2 Back-End Functional Requirements

3.1.2.1 - Data Administration (Asim Amjad)

The data administration functional area must provide the security, efficiency, and flexibility required for maintaining the data. This functional area will be capable of controlling the general access to the system and maintaining the information contained in the data tables. ResearchLink must use sixteen tables to create the database.

3.1.2.1.1 ResearchLink must use MySQL database to store all information for administrators, faculty members, and students. The following functional requirements must be met for each database table:

   3.1.2.1.1.1 academic_subjects Table
   The academic_subjects table will hold the information of the subjects offered by the ODU. Each subject must have an auto incremental id. The following functional requirements must be met:
   1. The attribute id must have a primary key with auto increments.
   2. The attribute name must be non-null.
   3. The attribute department_id must be a foreign key to the departments table on id attribute.
   4. All attributes in the academic_subjects must be non-null.
   5. The following attributes must appear in the academic_subjects table.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
<th>Not-Null</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Int(11)</td>
<td>Yes</td>
<td>Primary key</td>
<td>ID for subjects</td>
</tr>
<tr>
<td>name</td>
<td>Varchar(45)</td>
<td>Yes</td>
<td></td>
<td>Name of the subject</td>
</tr>
</tbody>
</table>
3.1.2.1.1.2 categories Table
The categories table will hold the information of the different disciplines for one field. For example, Computer Science includes software engineering, application development, or artificial intelligence. The following functional requirements must be met:
1. The attribute id must have a primary key with auto increments.
2. The attribute name must be non-null.
3. The attribute academic_subject must be a foreign key to the academic_subjects table on id attribute.
4. All attributes in the category table must be non-null.
5. The following attributes must appear in the categories table.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
<th>Not-Null</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Int(11)</td>
<td>Yes</td>
<td>Primary key</td>
<td>ID for categories</td>
</tr>
<tr>
<td>name</td>
<td>Varchar(45)</td>
<td>Yes</td>
<td></td>
<td>Name of the category</td>
</tr>
<tr>
<td>academic_subject</td>
<td>Int(11)</td>
<td>Yes</td>
<td>Foreign key with departments</td>
<td>ID for each subject</td>
</tr>
</tbody>
</table>

Table 2. categories Table

3.1.2.1.1.3 courses Table
The courses table will represent the details of the courses offered by ODU. Each course id must have auto increments. The following functional requirements must be met:
1. The attribute idcourses must have a primary key with auto increments.
2. The attribute course_number must accept a default value null.
3. The attribute academic_subject must be a foreign key to the academic_subjects table on id attribute.
4. The following attributes must appear in the courses table.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
<th>Not-Null</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
</table>
### Table 3. courses Table

#### 3.1.2.1.1.4 courses_taken Table

The courses_taken table will hold the information of courses enrolled by students. There must be an auto increment id in this table. This table will use users and courses table data. The following functional requirements must be met:

1. The attribute id must have a primary key with auto increments.
2. The attribute user_id must be a foreign key to the user_id table on id attribute.
3. The attribute course_id must be a foreign key to the courses table on idcourses attribute.
4. The following attributes must appear in the courses_taken table.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
<th>Not-Null</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Int(11)</td>
<td>Yes</td>
<td>Primary key</td>
<td>ID for courses_taken</td>
</tr>
<tr>
<td>user_id</td>
<td>Int(11)</td>
<td>No</td>
<td>Foreign key with users</td>
<td>ID for user</td>
</tr>
<tr>
<td>course_id</td>
<td>Int(11)</td>
<td>No</td>
<td>Foreign key with courses</td>
<td>ID for course</td>
</tr>
</tbody>
</table>

### Table 4. courses_taken Table

#### 3.1.2.1.1.5 courses_taught Table

The courses_taught table will hold the information of courses taught by faculty. There must be an auto increment id in this table. This table will use users and courses table data. The following functional requirements must be met:

1. The attribute id must have a primary key with auto increments.
2. The attribute user_id must be a foreign key to the user_id table on id attribute.
3. The attribute course_id must be a foreign key to the courses table on idcourses attribute.
4. The following attributes must appear in the courses_taught table.
<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
<th>Not-Null</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Int(11)</td>
<td>Yes</td>
<td>Primary key</td>
<td>ID for courses_taught</td>
</tr>
<tr>
<td>user_id</td>
<td>Int(11)</td>
<td>No</td>
<td>Foreign key with users</td>
<td>ID for user</td>
</tr>
<tr>
<td>course_id</td>
<td>Int(11)</td>
<td>No</td>
<td>Foreign key with courses</td>
<td>ID for course</td>
</tr>
</tbody>
</table>

Table 5. courses_taught Table

3.1.2.1.1.6 departments Table

The department table will hold the information of the ODU departments. There must be an auto increment id in this table. This table will use users and courses table data. The following functional requirements must be met:

1. The attribute id must have an id with auto increments.
2. The attribute name must have a department name.
3. The following attributes must appear in the departments table.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
<th>Not-Null</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Int(11)</td>
<td>Yes</td>
<td>Primary key</td>
<td>ID for departments</td>
</tr>
<tr>
<td>name</td>
<td>Varchar(45)</td>
<td>Yes</td>
<td></td>
<td>Department Name</td>
</tr>
</tbody>
</table>

Table 6. departments Table

3.1.2.1.1.7 interest_areas Table

The interest_areas table will hold the information of the interest for a student, or faculty member. There must be an auto increment id in this table. This table will use users table data. The following functional requirements must be met:

1. The attribute id must have a primary key with auto increments.
2. The attribute user_id must be a foreign key to the user table on id attribute.
3. The attribute category_id must be a foreign key to the category table on category_id attribute.
4. The following attributes must appear in the interest_areas table.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
<th>Not-Null</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Int(11)</td>
<td>Yes</td>
<td>Primary key</td>
<td>ID for interest</td>
</tr>
<tr>
<td>user_id</td>
<td>Int(11)</td>
<td>Yes</td>
<td>Foreign key with users</td>
<td>ID for a user</td>
</tr>
</tbody>
</table>
### 3.1.2.1.8 migrations Table

The migrations table will create a version control of the ResearchLink database. This table will keep a record of how ResearchLink database was created and altered over time. The changes to the database will use PHP for migration table instead of SQL. The following functional requirements must be met:

1. The attribute migration must have a non-null value.
2. The attribute batch must have a non-null value.
3. The following attributes must appear in the migrations table.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
<th>Not-Null</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Migration</td>
<td>Varchar(255)</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Batch</td>
<td>Int(11)</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 3.1.2.1.9 notifications Table

The notifications table will send notifications to faculty and students. An algorithm, SendNotification will notify the related users after comparing the deadline date with the current date. SendNotification will be tracked the notifications through a Boolean value which will be a false value by default. Another algorithm, AddNotifications, will add a record in the notification table after notifying the users. The following functional requirements must be met:

1. The attribute id must be a primary key with auto increments.
2. The attribute user_id must have a FK with users table on id attribute.
3. The attribute type_of_notification must accept a default value null.
4. The attribute title_html must accept a default value null.
5. The attribute body_html must use Blob datatype to store the large html file with by accepting default value null.
6. The attribute is_read must use accept a default value null.
7. The attribute created_at must use accept a default value null.
8. The attribute updated_at must use accept a default value null.
9. The following attributes must appear in the notifications table.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
<th>Not-Null</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
</table>

### Table 9. Notifications Table

3.1.2.1.1.10 password_resets Table

The password_resets table will update the password for users in case someone forgets a password. The following functional requirements must be met:

1. The attribute email must be non-null.
2. The attribute token must be used for token-based authentication system with a prohibited null values. This attribute must update the current password credentials too.
3. The attribute created_at must do not enter Null values.
4. The following attributes must appear in the password_resets table.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
<th>Not-Null</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>email</td>
<td>Varchar(255)</td>
<td>Yes</td>
<td></td>
<td>Emails used as user ID</td>
</tr>
<tr>
<td>token</td>
<td>Varchar(255)</td>
<td>Yes</td>
<td></td>
<td>Used for authentication</td>
</tr>
<tr>
<td>created_at</td>
<td>Timestamp</td>
<td>Yes</td>
<td></td>
<td>Record time for updated password</td>
</tr>
</tbody>
</table>

Table 10. password_resets Table

3.1.2.1.1.11 profile Table
The profile table will create profiles for users - faculty, students, and research organizations. The following functional requirements must be met:

1. The attribute user_id must be a primary key with auto increments. This attribute must have a FK with users table on id attribute.
2. The attribute address must accept default value null.
3. The attribute city must accept default value null.
4. The attribute zipcode must accept default value null.
5. The attribute phone must accept default value null.
6. The attribute user_type must accept default value null.
7. The attribute image must accept default value null.
8. The attribute department must accept default value null.
9. The attribute zipcode must accept default value null.
10. The attribute phone must accept default value null.
11. The attribute user_type must accept default value null.
12. The attribute image must accept default value null.
13. The attribute department must accept default value null.
14. The following attributes must appear in the profile table.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
<th>Not-Null</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user_id</td>
<td>Int(10)</td>
<td>Yes</td>
<td>Primary Key</td>
<td>ID for users</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Foreign key</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>with users</td>
<td></td>
</tr>
<tr>
<td>address</td>
<td>Varchar(45)</td>
<td>No</td>
<td></td>
<td>Street address of user</td>
</tr>
<tr>
<td>city</td>
<td>Varchar(50)</td>
<td>No</td>
<td></td>
<td>City of user</td>
</tr>
<tr>
<td>state</td>
<td>Varchar(45)</td>
<td>No</td>
<td></td>
<td>State of user</td>
</tr>
<tr>
<td>zipcode</td>
<td>Varchar(45)</td>
<td>No</td>
<td></td>
<td>Zip code of user</td>
</tr>
<tr>
<td>phone</td>
<td>Varchar(45)</td>
<td>No</td>
<td></td>
<td>Phone number of user</td>
</tr>
<tr>
<td>user_type</td>
<td>Varchar(45)</td>
<td>No</td>
<td></td>
<td>User type (faculty or student)</td>
</tr>
<tr>
<td>image_name</td>
<td>Varchar(50)</td>
<td>No</td>
<td></td>
<td>User photo</td>
</tr>
<tr>
<td>major</td>
<td>Varchar(50)</td>
<td>No</td>
<td></td>
<td>Only student major</td>
</tr>
<tr>
<td>department</td>
<td>Varchar(45)</td>
<td>No</td>
<td></td>
<td>User department</td>
</tr>
</tbody>
</table>
### Table 11. profile Table

#### 3.1.2.1.1.12 research_agencies Table

The research_agencies table will provide information for the research agency. The following functional requirements must be met:

1. The attribute id must be a primary key with auto increments.
2. The attribute name must accept null values.
3. The attribute image_name must accept null values.
4. The following attributes must appear in the research_agencies table.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
<th>Not-Null</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Int(11)</td>
<td>Yes</td>
<td>Primary key</td>
<td>Identification for research agency</td>
</tr>
<tr>
<td>name</td>
<td>Varchar(60</td>
<td>No</td>
<td></td>
<td>Name of the research agency</td>
</tr>
<tr>
<td>image_name</td>
<td>Varchar(45</td>
<td>No</td>
<td></td>
<td>Image or logo of the research agency</td>
</tr>
</tbody>
</table>

### Table 12. research_agencies Table

#### 3.1.2.1.1.13 research_contact Table

The research_contact table will provide contact information for the research organization in relation to a research opportunity. The following functional requirements must be met:

1. The attribute id must be a primary key with auto increments. This attribute must have FK with research_agency on id attribute.
2. The attribute opportunity_id must have FK with research_opportunities on id attribute.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
3. The attribute first_name must accept null values.
4. The attribute last_name must accept null values.
5. The attribute email must accept null values.
6. The attribute phone must accept null values.
7. The following attributes must appear in the research_contact table.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
<th>Not-Null</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Int(11)</td>
<td>Yes</td>
<td>Primary key</td>
<td>ID for research agency</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Foreign key with research_agency</td>
<td></td>
</tr>
<tr>
<td>opportunity_id</td>
<td>Int(11)</td>
<td>No</td>
<td></td>
<td>ID for an opportunity</td>
</tr>
<tr>
<td>first_name</td>
<td>Varchar(45)</td>
<td>No</td>
<td></td>
<td>Agency’s employee First Name</td>
</tr>
<tr>
<td>last_name</td>
<td>Varchar(45)</td>
<td>No</td>
<td></td>
<td>Agency’s employee second name</td>
</tr>
<tr>
<td>email</td>
<td>Varchar(45)</td>
<td>No</td>
<td></td>
<td>Agency’s email address</td>
</tr>
<tr>
<td>phone</td>
<td>Varchar(45)</td>
<td>No</td>
<td></td>
<td>Agency’s phone number</td>
</tr>
</tbody>
</table>

Table 13. research_contact Table

3.1.2.1.1.14 research_opportunity Table
The research opportunity table will create opportunity records for research opportunities entered by faculty and administrators. The following functional requirements must be met:
1. The attribute id must be primary key with auto increments.
2. The attribute agency_id must be a FK with research_agencies table on id attribute.
3. The attribute user_id must be a FK with users table on id attribute.
4. The attribute title must accept default value null.
5. The attribute description must accept null values.
6. The attribute category_id must be a FK with categories table on id attribute.
7. The attribute department_id must have a FK with department table on id attribute.
8. The attribute distance_learning must accept default value null.
9. The attribute type must accept default value null.
10. The attribute paid must accept default value null.
11. The attribute payment_amount must accept default value null.
12. The attribute app_start must accept default value null.
13. The attribute app_end must accept default value null.
14. The attribute research_start must accept default value null.
15. The attribute research_end must accept default value null.
16. The attribute created_by must accept default value null.
17. The attribute created_at must accept default value null.
18. The attribute updated_at must accept default value null.
19. The attribute expired must accept default value null.
20. The following attributes must appear in the research_opportunity table.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
<th>Not-Null</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Int(11)</td>
<td>Yes</td>
<td>Primary key</td>
<td>ID for research opportunity</td>
</tr>
<tr>
<td>agency_id</td>
<td>Int(11)</td>
<td>No</td>
<td>Foreign key with agencies</td>
<td>ID for research agency</td>
</tr>
<tr>
<td>user_id</td>
<td>Int(11)</td>
<td>No</td>
<td>Foreign key with users</td>
<td>ID for a user either a research agency or faculty member</td>
</tr>
<tr>
<td>title</td>
<td>Varchar(100 )</td>
<td>No</td>
<td></td>
<td>Title of the opportunity</td>
</tr>
<tr>
<td>description</td>
<td>Blob</td>
<td>No</td>
<td></td>
<td>Detail about a research opportunity</td>
</tr>
<tr>
<td>category_id</td>
<td>Int(11)</td>
<td>No</td>
<td>Foreign key with categories</td>
<td>ID for a category</td>
</tr>
<tr>
<td>department_id</td>
<td>Int(11)</td>
<td>No</td>
<td>Foreign key with departments</td>
<td>ID for a department</td>
</tr>
<tr>
<td>distance_learning</td>
<td>Tinyint(4)</td>
<td>No</td>
<td></td>
<td>Teach/learn from distance</td>
</tr>
<tr>
<td>type</td>
<td>Varchar(45)</td>
<td>No</td>
<td></td>
<td>Type of opportunity</td>
</tr>
<tr>
<td>paid</td>
<td>Tinyint(4)</td>
<td>No</td>
<td></td>
<td>Opportunity paid/unpaid</td>
</tr>
<tr>
<td>Attributes</td>
<td>Type</td>
<td>Not-Null</td>
<td>Key</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------</td>
<td>----------</td>
<td>------------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>id</td>
<td>Int(11)</td>
<td>Yes</td>
<td>Primary key</td>
<td>Unique ID</td>
</tr>
<tr>
<td>opportunity_id</td>
<td>Int(11)</td>
<td>No</td>
<td>Foreign key with research_opportunity</td>
<td>ID for an opportunity</td>
</tr>
<tr>
<td>user_id</td>
<td>Int(11)</td>
<td>No</td>
<td>Foreign key with users</td>
<td>ID for users</td>
</tr>
<tr>
<td>applied</td>
<td>Tinyint(4)</td>
<td>No</td>
<td></td>
<td>Track if student has applied to opportunity</td>
</tr>
</tbody>
</table>

Table 15. Saved_opportunities Table

3.1.2.1.16 users Table
The users table will store users in the Research Link database. The following functional requirements must be met:

1. The attribute id must be a primary key with auto increments.
2. The attribute first_name must not accept null values.
3. The attribute last_name must accept null values.
4. The attribute last_name must accept null values.
5. The attribute email must not accept null values.
6. The attribute password must have the following attributes.
   a. Password field must not accept null values.
   b. Password field must consist of at least 8 characters.
   c. Password field must have at least one number.
   d. Password field must have at least one alphabet character.
2. The attribute remember_token must accept null values.
3. The attribute confirmation_code must accept null values.
4. The attribute created_at must accept null values.
5. The attribute updated_at must accept null values.
6. The following attributes must appear in the users table:

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
<th>Not-Nul</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Int(10)</td>
<td>Yes</td>
<td>Primary key</td>
<td>ID for users</td>
</tr>
<tr>
<td>first_name</td>
<td>Varchar(255)</td>
<td>Yes</td>
<td></td>
<td>First name of user</td>
</tr>
<tr>
<td>last_name</td>
<td>Varchar(255)</td>
<td>No</td>
<td></td>
<td>Last name of user</td>
</tr>
<tr>
<td>email</td>
<td>Varchar(255)</td>
<td>Yes</td>
<td></td>
<td>Email address of user</td>
</tr>
<tr>
<td>password</td>
<td>Varchar(255)</td>
<td>Yes</td>
<td></td>
<td>Password for user</td>
</tr>
<tr>
<td>remember_token</td>
<td>Varchar(100)</td>
<td>No</td>
<td></td>
<td>Authentication for password</td>
</tr>
<tr>
<td>confirmation_code</td>
<td>Varchar(100)</td>
<td>No</td>
<td></td>
<td>Send code via email</td>
</tr>
<tr>
<td>confirmed</td>
<td>Tinyint(4)</td>
<td>No</td>
<td></td>
<td>Code status after confirmation</td>
</tr>
<tr>
<td>created_at</td>
<td>Timestamp</td>
<td>No</td>
<td></td>
<td>Account creation date</td>
</tr>
</tbody>
</table>
3.1.2.1.1.17 news_highlight Table

The news highlight table will create news highlight records for display in the looping newsfeed. The following functional requirements must be met:

1. The attribute id must be primary key with auto increments.
2. The attribute user_id must be a FK with users table on id attribute.
3. The attribute title must accept default value null.
4. The attribute description must accept null values.
5. The attribute category_id must be a FK with categories table on id attribute.
6. The attribute department_id must have a FK with department table on id attribute.
7. The attribute public must accept default value null.
8. The attribute payment_amount must accept default value null.
9. The attribute post_date must accept default value null.
10. The attribute expiration_date must accept default value null.
11. The attribute expired must accept default value null.
12. The following attributes must appear in the news_highlight table.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Type</th>
<th>Not-Null</th>
<th>Key</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Int(11)</td>
<td>Yes</td>
<td>Primary key</td>
<td>ID for news highlight</td>
</tr>
<tr>
<td>user_id</td>
<td>Int(11)</td>
<td>No</td>
<td>Foreign key with users</td>
<td>ID for faculty member</td>
</tr>
<tr>
<td>title</td>
<td>Varchar(100)</td>
<td>No</td>
<td></td>
<td>Title of the news highlight</td>
</tr>
<tr>
<td>description</td>
<td>Blob</td>
<td>No</td>
<td></td>
<td>Detail about news highlight</td>
</tr>
<tr>
<td>category_id</td>
<td>Int(11)</td>
<td>No</td>
<td>Foreign key with categories</td>
<td>ID for a category</td>
</tr>
<tr>
<td>department_id</td>
<td>Int(11)</td>
<td>No</td>
<td>Foreign key with departments</td>
<td>ID for a department</td>
</tr>
</tbody>
</table>
### 3.1.2.2 - Notification System (John Raha)

This notification system contains both a web-based system and an email system which sends notifications to registered users. Notifications include messages related to research opportunity creation, account status changes, and information related to a user’s department. The following functional requirements must be met:

1. Trigger a notification to students when a new opportunity has been created matching the student's profile list of interests.
   a. The user must have selected profile interests.
   b. The opportunity must have selected keywords.
   c. The user must match a minimum of three keywords.
   d. The notification will use a predefined template.

2. Trigger a notification to department members with department news.
   a. The user’s selected department must match the message sender’s department.
   b. The notification must not be null.
   c. The notification will be custom.

3. The system shall allow administrators to send notifications to all registered users.
   a. The notification must not be null.
   b. The notification will be custom.

4. Website notifications will contain a text message summarizing the triggering event.
   a. The notification must not be null.
   b. The notification will be either custom or use a predefined template.

5. Website notifications will be available in the user’s message center.

---

<table>
<thead>
<tr>
<th>post_date</th>
<th>Date</th>
<th>No</th>
<th>News highlight post date</th>
</tr>
</thead>
<tbody>
<tr>
<td>expiration_date</td>
<td>Date</td>
<td>No</td>
<td>News highlight expiration date</td>
</tr>
<tr>
<td>expired</td>
<td>Tinyint(4)</td>
<td>No</td>
<td>News highlight expired/not expired</td>
</tr>
</tbody>
</table>

Table 17. news_highlight Table
a. Website notifications may be marked as read.
b. Website notifications marked as read will be deleted.

6. The system shall trigger an email notification when there is also a website notification.
   a. The user must have email notifications activated.

7. Notifications shall be sent the registered user's ODU email address.

8. All notifications sent to a user’s email are the same notifications displayed on the website.

9. Users may turn off email notifications at any time.

3.1.2.3 - Research Opportunity Archival Feature (Dinah Watkins)

This calendar based archival feature will assist in the toggle of the expired field of opportunity records that have passed the designated opportunity end date and removed from being accessible by active searches. The following functional requirements must be met:

1. Prepopulate the database with fake opportunities to be used during prototype demonstration
2. Provide the capability to auto-archive an opportunity when the opportunity end date has passed
   a. Algorithm is triggered at 3AM daily to toggle records as expired
   b. Calendar based
   c. The below algorithm illustrates the logic behind this feature:

3. Demonstrated through modification of the system date and time
3.1.2.4 - News/Highlights Archival Feature (Dinah Watkins)

This calendar based archival feature will assist in the toggle of the expired field of news/highlight records that have passed the designated expiration date. The following functional requirements must be met:

1. Provide the capability to auto-archive an opportunity when the expiration date has passed
   a. Algorithm is triggered at 3AM daily to toggle records as expired
   b. Calendar based
   c. The below algorithm illustrates the logic behind this feature:

   ![Algorithm Diagram]

2. Demonstrated through modification of the system date and time

3.1.2.5 - New User Page (Matthew Redenius)

This is a web page use to store new user information in the database. The following functional requirements must be met:

1. Prepopulate the database with fake email addresses for account type authentication to be used during prototype demonstration
   a. Faculty email addresses
   b. Student email addresses
2. Must save string in listed text fields to database table “users”
   a. First Name
   b. Last Name
c. Username
   i. Must query “users” table to check if username already exists
   ii. The below algorithm illustrates the logic behind this feature:

   ![Diagram]

   d. Password

   e. Email
      i. Must query “user” table to check if email already exists
      ii. If faculty usertype is chosen, database must be queried to see if faculty email is approved faculty email

3. Faculty user types will be differentiated from other users types
   a. Email address will be verified by confirming it exists in the pre-populated mock list of approved faculty emails.
   b. The below algorithm illustrates the logic behind this feature:
### 3.1.2.6- Search Opportunities Page (Matthew Redenius)

This is a web page that queries “saved opportunities” table in the database to display opportunity data. The following functional requirements must be met:

1. Must query “saved opportunities” table based on user inputted keyword search criteria for:
   a. Opportunity title
   b. Opportunity type
   c. Opportunity Description

2. Must query “saved opportunities” table based on user selection for:
   a. Opportunity paid
   b. Opportunity non-paid
   c. Application end
   d. Opportunity start

3. The below algorithm illustrates the logic behind this feature:
User fills keyword search block

User presses search

Are there opportunities currently displayed? Yes

Remove currently displayed opportunities from display

No

Query database with user input

Yes

Display the results

Did query return any results? Yes

No

Display error message