**Team**: Michael Bernabei, Akash Devgun, Jose Vieitez

**Title**:  Mentor Customer Relationship Management (CRM) System

**Project Summary**: The purpose of the Mentor CRM is to store data on mentors and use the information stored in the database as means to determine how to best pair mentors with companies.

As of today there isn’t a worthwhile mentor CRM system. Our plan is to build a worthy mentor system that could see real world use.

The system would allow someone managing a mentor network to manage and update mentor information, track how often they mentor along with ratings and feedback each mentor gets. The system will include: profiles, ratings, search, and sorting capabilities.

**Project Requirements**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Business Requirements** | | | |  |
| **ID** | **Requirement** | **Topic Area** | **User** | **Priority** |
| **BR-001** | Mentors need to provide at least one email linked to a LinkedIn profile. | Authentication | Mentor | Critical |
| **BR-002** | Mentors must provide at least one valid form of contact information. | Profile | System | Critical |
| **BR-003** | Companies must provide contact information |  |  | Critical |
| **BR-004** | Mentor profiles need to have a rating field. | Profile |  | Critical |
| **BR-005** | Mentor profiles need to have a comment box. | Profile |  | Critical |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **User Requirements** | | | | |
| **ID** | **Requirement** | **Topic Area** | **User** | **Priority** |
| **UR-001** | As an admin, I need to be able to see settings to give or restrict access to the modification of all mentor profiles in the system for other Admin roles. | Mentor Profile | Admin | High |
| **UR-002** | The system shall support user roles, e.g. Admin, ViewLimited, and ViewOnly users. | Authentication |  | Critical |
| **UR-003** | As an Admin, I need to be able modify all other mentor profiles in the system. |  | Admin | Critical |
| **UR-004** | As an Admin, I need to be able to log in and out of the system | Authentication |  | Critical |
| **UR-005** | As an Admin, I need to be able to add other Admins | Authentication | Admin |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Functional Requirements** | | | | |
| **ID** | **Requirement** | **Topic Area** | **User** | **Priority** |
| **FR-001** | The system must add information from a mentor’s LinkedIn profile. | Mentor Profile | System | Critical |
| **FR-002** | A mentor’s profile must be stored in the database. | Mentor Profile | System, DB | Critical |
| **FR-003** | A company’s name and contact information must be stored in the database | Company Profile | System, DB | Critical |
| **FR-004** | A mentor's mentoring history must be stored in the database. |  | System, DB, Mentor | Critical |
| **FR-005** | Create Company and Mentor Meeting |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Non-Functional Requirements** | | | |  |
| **ID** | **Requirement** | **Topic Area** | **User** | **Priority** |
| **NFR-001** | The system should retrieve a LinkedIn Profile in under 1 minute. | Mentor Profile | System | Medium |
| **NFR-002** | The system should do all searching and sorting in under 7 seconds |  |  | High |
| **NFR-003** |  |  |  |  |

**Use Cases**:

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | FR-001 | | |
| Use Case Name: | Store a mentor’s LinkedIn profile | | |
| Created By: | Michael Bernabei | Last Updated By: |  |
| Date Created: | Feb 22, 2016 | Date Last Updated: |  |

|  |  |
| --- | --- |
| Actor: | Administrator, Full Contact API, Database |
| Description: | Store a mentor’s linkedIn in the database. |
| Preconditions: | * System should have parsed the comma separated email address list. * Mentor should have a valid LinkedIn account. |
| Postconditions: | * Mentor’s LinkedIn information is stored in the database. |
| Priority: | Critical |
| Frequency of Use: | Monthly |
| Normal Course of Events: | 1. The administrator provides a list of mentors to upload to the system. 2. The system parses the email addresses. 3. The system calls the Full Contact API passing the mentor email as a parameter. 4. The Full Contact API returns its search results to the system. 5. The results from the Full Contact API are parsed and stored in the database. |
| Alternative Courses: | * LinkedIn information is not available * The Administrator will be alerted to the issue * Contact Must be made with the Mentor in order to retrieve the required information to be stored in the database. |
| Exceptions: | * Mentor’s LinkedIn information is not available. * Administrator login fails. * Full Contact API rejects request from the system. * Full Contact API doesn’t find information on mentors. * System fails to parse the mentor email addresses. |
| Includes: | * Get Full Contact Results |
| Extends: | Store Mentor Profile |
| Special Requirements: | * System needs to have access to Full Contact API |
| Assumptions: | * Mentor has a valid LinkedIn account |
| Notes and Issues: | A developer relationship is required for successful use of the Full Contact API. A developer key must be received from Full Contact before the system can use their API. |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | FR-005 | | |
| Use Case Name: | Save Company and Mentor Meeting | | |
| Created By: | Michael Bernabei | Last Updated By: |  |
| Date Created: | Feb 22, 2016 | Date Last Updated: |  |

|  |  |
| --- | --- |
| Actor: | Administrator |
| Description: | Save a meeting with a mentor and a company in the database. |
| Preconditions: | * Administrator has created a link between a given mentor and a company. * The Mentor, Company, and Meeting information are received on the backend. |
| Postconditions: | * Meeting interaction between the company and the mentor are stored in the database. |
| Priority: | Critical |
| Frequency of Use: | Daily |
| Normal Course of Events: | 1. The administrator creates a meeting between a mentor and a company. 2. The company, mentor, and meeting information are checked for time conflicts. 3. If no time conflicts exist the information is stored in the database. |
| Alternative Courses: | * Time conflicts are found and the administrator is alerted. * Mentor and companies are alerted about time conflicts. |
| Exceptions: | * Time conflict exception |
| Includes: | * Create a company and mentor meeting * Check for time conflicts |
| Special Requirements: | Human in the loop |
| Notes and Issues: | While meetings are saved in the database a human in the loop is still required. Consequently a user forgetting to create a meeting can cause conflicts in the system. |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | FR-004 | | |
| Use Case Name: | A mentor’s mentoring history must be stored in the database. | | |
| Created By: | Michael Bernabei | Last Updated By: |  |
| Date Created: | Feb 22, 2016 | Date Last Updated: |  |

|  |  |
| --- | --- |
| Actor: | Administrator, Company |
| Description: | A mentor’s meeting interactions history must be shown |
| Preconditions: | * Mentor has mentored at least one company |
| Postconditions: | * Mentor’s mentoring history is stored in the database |
| Priority: | High |
| Frequency of Use: | Weekly |
| Normal Course of Events: | 1. Mentoring confirmation from the company and mentor is received. 2. The administrator will then create a link between the mentor and company indicating successful mentoring occurred. 3. The system will then store the link in the database. |
| Alternative Courses: | * No mentoring confirmation is received. |
| Exceptions: | * Company or Mentor information is not found in the database. |
| Includes: | * Confirm the mentor mentored a company * Get mentor history * Get company history |
| Special Requirements: | Human in the loop |
| Assumptions: | * Mentor has mentored at least one company * Company has been mentored by at least one mentor |
| Notes and Issues: | A human in the loop is required to ensure that a company successfully mentored a company. |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UR-003 | | |
| Use Case Name: | Admin editing usage rights | | |
| Created By: | Michael Bernabei | Last Updated By: |  |
| Date Created: | Feb 22, 2016 | Date Last Updated: |  |

|  |  |
| --- | --- |
| Actor: | Admin, Mentor |
| Description: | A user that signs in with administrative user rights will be able to edit all mentor profiles. |
| Preconditions: | * There is at least one mentor stored in the database |
| Postconditions: | * At least one mentor profile has been edited by the administrator |
| Priority: | Critical |
| Frequency of Use: | Weekly |
| Normal Course of Events: | 1. A user signs in using a valid administrator account. 2. The administrator sees a settings icon which allows the user to edit all other mentor accounts in the system. |
| Alternative Courses: | * User fails to successfully sign in as administrator |
| Exceptions: | * There aren’t any mentor profiles stored in the database |
| Includes: | * Display Administrative Profile View |
| Extends: | * Logon to System |
| Special Requirements: | None |
| Assumptions: | * A user has already successfully created an Administrative account. |
| Notes and Issues: | None |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC-009 | | |
| Use Case Name: | Searching for which mentors have mentored a specific company | | |
| Description: | The Admin can search for mentors that have mentored a specific company in the past that is similar to a new company, in order to also match them together. | | |
| Created By: | Jose | Last Updated By: |  |
| Date Created: | March 1, 2016 | Date Last Updated: |  |

|  |  |
| --- | --- |
| Actor: | Admin |
|  |  |
| Preconditions: | Admin is logged in. Database has mentors and past companies they have mentored, stored. |
| Postconditions: | Admin receives a list of mentors that have mentored a specific company in the past, or confirmation that there are no mentors who have done so. |
| Frequency of Use: | Multiple times per hour for 6 specific weeks per year, and then weekly throughout year. |
| Normal Course of Events: | |  |  | | --- | --- | | **Actor Action** | **System Response** | | 1.  Click Search bar |  | | 2. Type Company name & click search | List of mentors displayed that have mentored this company | |  |  | |
| Variations: | 3. Admin clicks on a mentor profile for more information |
| Notes and Issues: |  |
| Developer Notes: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC-010 | | |
| Use Case Name: | Search for Mentor by Name | | |
| Description: | When an Admin knows exactly who to search for to find information about them, they can search by the name of that mentor | | |
| Created By: | Jose | Last Updated By: |  |
| Date Created: | March 1, 2016 | Date Last Updated: |  |

|  |  |
| --- | --- |
| Actor: | Admin |
|  |  |
| Preconditions: | Admin is logged in. Database has populated mentor profiles |
| Postconditions: | Admin receives the profile information about a specific mentor |
| Frequency of Use: | Multiple times per hour for 6 specific weeks per year, and then daily throughout year. |
| Normal Course of Events: | |  |  | | --- | --- | | **Actor Action** | **System Response** | | 1.  Click Search bar |  | | 2. Type mentor name & click search | List of mentors displayed that match name | | 3. Click name of mentor | Displays the full profile of the selected mentor | |
| Variations: | 4. Admin edits the inputted name due to error |
| Notes and Issues: |  |
| Developer Notes: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC-011 | | |
| Use Case Name: | Manually input Mentor Data | | |
| Description: | If a Mentor does not have a Linkedin profile or part of their profile information is missing, the Admin has to input this data manually | | |
| Created By: | Jose | Last Updated By: |  |
| Date Created: | March 1, 2016 | Date Last Updated: |  |

|  |  |
| --- | --- |
| Actor: | Admin |
|  |  |
| Preconditions: | Admin is logged in. Admin has clicked “New Mentor” |
| Postconditions: | The mentor  profile now has a field populated which is usually filled in automatically |
| Frequency of Use: | Multiple times per hour early on until the database is populated, and then once per week |
| Normal Course of Events: | |  |  | | --- | --- | | **Actor Action** | **System Response** | | 1.  Click empty text field |  | | 2. Type appropriate information for that field |  | | 3. Click “Save” | Message saying “Profile saved” appears | |
| Variations: | 3. Admin clicks on multiple other fields and edits them |
| Notes and Issues: |  |
| Developer Notes: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC-012 | | |
| Use Case Name: | Sort Mentors | | |
| Description: | While viewing a list of mentors, sorting and ranking them by a variety of options, including ranking, recent update, activeness | | |
| Created By: | Jose | Last Updated By: |  |
| Date Created: | March 1, 2016 | Date Last Updated: |  |

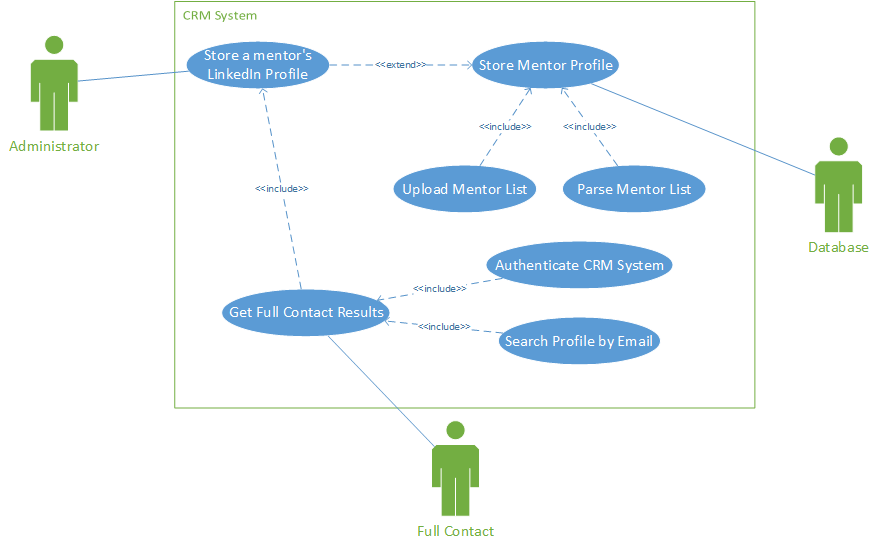
|  |  |
| --- | --- |
| Actor: | Admin |
|  |  |
| Preconditions: | Admin is logged in. Database has mentors. Admin is viewing a list of mentors. |
| Postconditions: | Admin receives a list of mentors that are rank sorted or ordered according to the desired query. |
| Frequency of Use: | Three times per week |
| Normal Course of Events: | |  |  | | --- | --- | | **Actor Action** | **System Response** | | 1.  Click Sort | Sort drop down options appear | | 2. Click Sort type | Mentor list is rearranged according to the sort type | |  |  | |
| Variations: | 3. Admin searches after sorting |
| Notes and Issues: |  |
| Developer Notes: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC-013 | | |
| Use Case Name: | Logging out | | |
| Description: | Admin logs out of the system to prevent further access until someone logs in. | | |
| Created By: | Jose | Last Updated By: |  |
| Date Created: | March 1, 2016 | Date Last Updated: |  |

|  |  |
| --- | --- |
| Actor: | Admin |
|  |  |
| Preconditions: | Admin has an existing account in the system. Admin logged in. |
| Postconditions: | Admin (and anyone else on that computer) no longer has access to the system, and is logged out. |
| Frequency of Use: | Once per month |
| Normal Course of Events: | |  |  | | --- | --- | | **Actor Action** | **System Response** | | 1.  Click Logout | User goes to the main login screen, prompting them to login in order to access system | |  |  | |
| Variations: |  |
| Notes and Issues: |  |
| Developer Notes: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC-014 | | |
| Use Case Name: | Adding a new Admin | | |
| Description: | Admin goes into settings and adds a new Admin to the system | | |
| Created By: | Jose | Last Updated By: |  |
| Date Created: | March 7, 2016 | Date Last Updated: |  |

|  |  |
| --- | --- |
| Actor: | Admin |
|  |  |
| Preconditions: | Admin has an existing account in the system. Admin logged in. Admin is in settings. |
| Postconditions: | There now will exist one additional Admin profile with admin rights as determined by the current Admin |
| Frequency of Use: | Once per year |
| Normal Course of Events: | |  |  | | --- | --- | | **Actor Action** | **System Response** | | 1.  Click Add New Admin | Shows page displaying info required to create an admin (email, password) | | 2. Admin fills out the information and clicks save | System saves a new admin with these new credentials | |
| Variations: | New admin account already exists: will show error |
| Notes and Issues: |  |
| Developer Notes: |  |

****

****

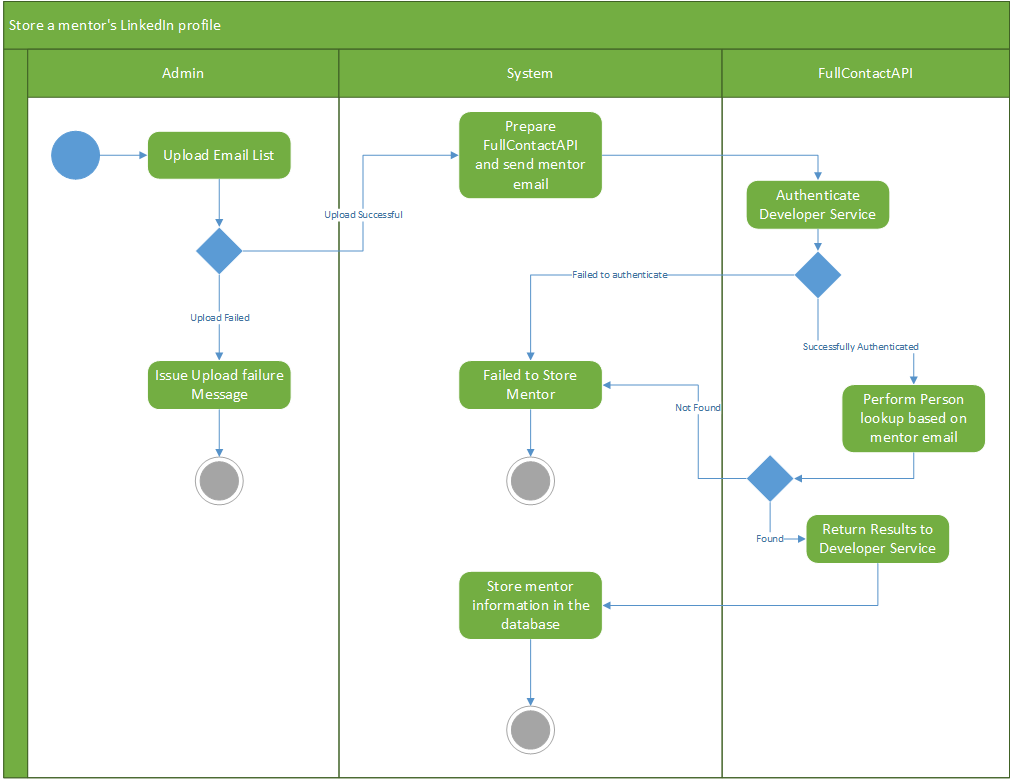
**Activity Diagrams**:

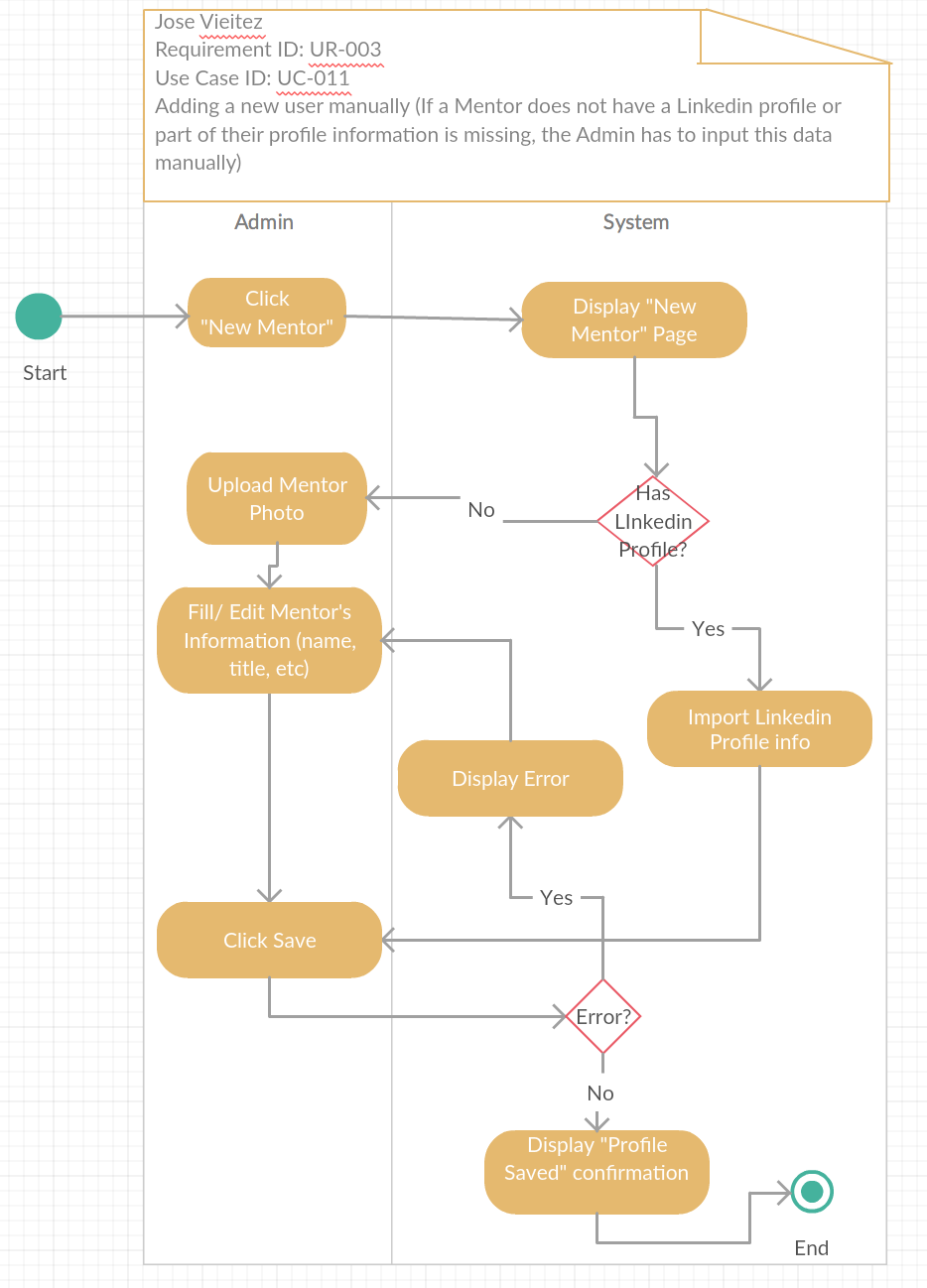
Requirement ID: FR-001

Use Case ID: FR-001

Store a mentor’s information from LinkedIn in the database

Michael Bernabei





***Data Storage:***

**Data Storage:** SQL Database

**Classes:**

* The MentorModel class will persist information about the mentors
* The CompanyModel class will persist information about the companies
* The User class will persist information about the system users

# Identified Entities/Subjects

* Mentor
* Mentor Contact Information
* Contact Method
* Status
* State
* Address
* Company
* Meeting
* Skill
* Feedback
* Rating
* Email
* User
* System Role

Mentors

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Mentor ID | int | A unique identification number given to all mentors in the system. |
| Lastname | char(n) | The last name of a mentor. |
| Middlename | char(n) | The middle name of a mentor. |
| Firstname | char(n) | The first name of a mentor. |
| LinkedIn Email Address | char(n) | A mentor’s LinkedIn email address. |
| Rating | foreign key | The rating value a mentor receives from a company. |

|  |  |  |
| --- | --- | --- |
| Feedback | foreign key | The feedback a mentor receives from a company they mentored. |
| Bio | text | The biographical information a mentor provides to the system. |
| Preferred Contact Method | foreign key | The Mentor’s preferred way to be contacted. |
| Contact Information | foreign key | A mentor’s contact information. |
| Status | foreign key | The current status of a mentor,  e.g. is the mentor busy, available, etc… |

Mentor Contact Information

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Contact ID | int | A unique contact ID issued for every mentor added to the system. |
| Mentor Name | foreign key | The mentor’s name. |
| Phone Number | int | The mentor’s number. |
| Address | foreign key | Contact address for a mentor. |
| Email | foreign key | The email address of the mentor. |

Status

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Status ID | int | A unique status ID for each status type. |
| Status Value | int | A status value indicating the current status of an entity. |

State

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| State ID | int | A unique ID for every state. |
| Name | char(n) | Name of the state. |
| Abbreviation | char(2) | State abbreviation. |

Address

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Address ID | int | The unique identification for addresses. |
| City Name | char(n) | Name of City in which the mentor resides. |
| Zip Code | int | Zip code. |
| State | foreign key | The state. |
| Street | char(n) | Street number. |

Company

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Company ID | int | Unique identification number which identifies individual companies. |
| Name | char(n) | Name of the company. |
| Address | foreign key | The address of the company. |

Meeting

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Meeting ID | int | Unique identification number which identifies individual meetings. |
| Mentor Name | foreign key | The name of the mentor attending the meeting. |
| Company Name | foreign key | The name of the company attending the meeting. |
| Date | datetime | The date of the meeting. |

Skill

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Skill ID | int | Unique identification number which identifies individual skills. |
| Skill Name | char(n) | Name of the skill. |

Feedback

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Feedback ID | int | Unique identification number which identifies individual feedback. |
| Feedback | char(n) | Feedback provided by a companies experience with a given mentor. |
| Mentor Name | foreign key | The name of the mentor for which the feedback is for. |
| Company Name | foreign key | Name of the company which provided the feedback |
| Date | datetime | The date for which the feedback was provided. |

Rating

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Rating ID | int | Unique identification number which identifies individual rating. |
| Rating Value | int | The rating value |

Email

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Email ID | int | The unique identification for email addresses. |
| Email Address | char(n) | Email addresses. |

User

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| User ID | int | Unique ID for system user names. |
| User Login Name | char(n) | The login name for a user. |

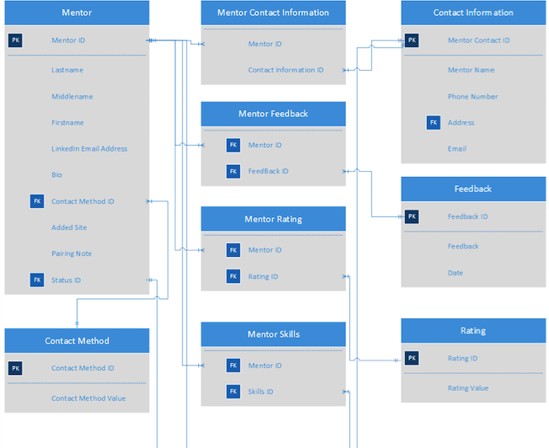
User Role

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Role ID | int | Unique ID for system role types. |
| Role Type | char(n) | The specific type of role a user of the system can have. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **Me** | **Mento** | **Cont** | **Stat** | **State** | **Addr** | **Com** | **Meet** | **Skill** | **Feedb** | **Rat** | **Em** | **Us** | **Sys** | | **nto** | **r** | **act** | **us** |  | **ess** | **pany** | **ing** |  | **ack** | **ing** | **ail** | **er** | **tem** | | **r** | **Conta** | **Meth** |  |  |  |  |  |  |  |  |  |  | **Rol** | |  | **ct** | **od** |  |  |  |  |  |  |  |  |  |  | **e** | |  | **Info.** |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Ment or**  **Ment or Cont act Info.**  **Cont act Meth od**  **Stat us**  **Stat e**  **Addr ess**  **Com pany**  **Meet ing**  **Skill**  **Feed back**  **Rati ng**  **Emai l**  **User**  **Syst em Role** |  | 1:N | 1:1 | 1:1 | 1:N |  |  |  | 1:N | 1:N | 1:N |  |  |  |
| 1:1 |  |  |  |  | 1:N |  |  |  |  |  | 1:N |  |  |
| 1:1 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1:1 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1:1 | 1:1 |  |  | 1:1 |  | 1:1 |  |  |  |  |  |  |  |
| 1:N | 1:N | 1:N |  | 1:N | 1:N |  | 1:N |  | 1:N | 1:N |  |  |  |
| 1:1 | 1:1 |  |  |  | 1:1 | 1:N |  |  |  |  |  |  |  |
| 1:N |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1:1 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1:N |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1:1 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  | 1:N |
|  |  |  |  |  |  |  |  |  |  |  |  | 1:N |  |

**Relation Addition:** 1:1 + 1:1 = 1:1 1:N + 1:N = M:N 1:1 + 1:N = 1:N

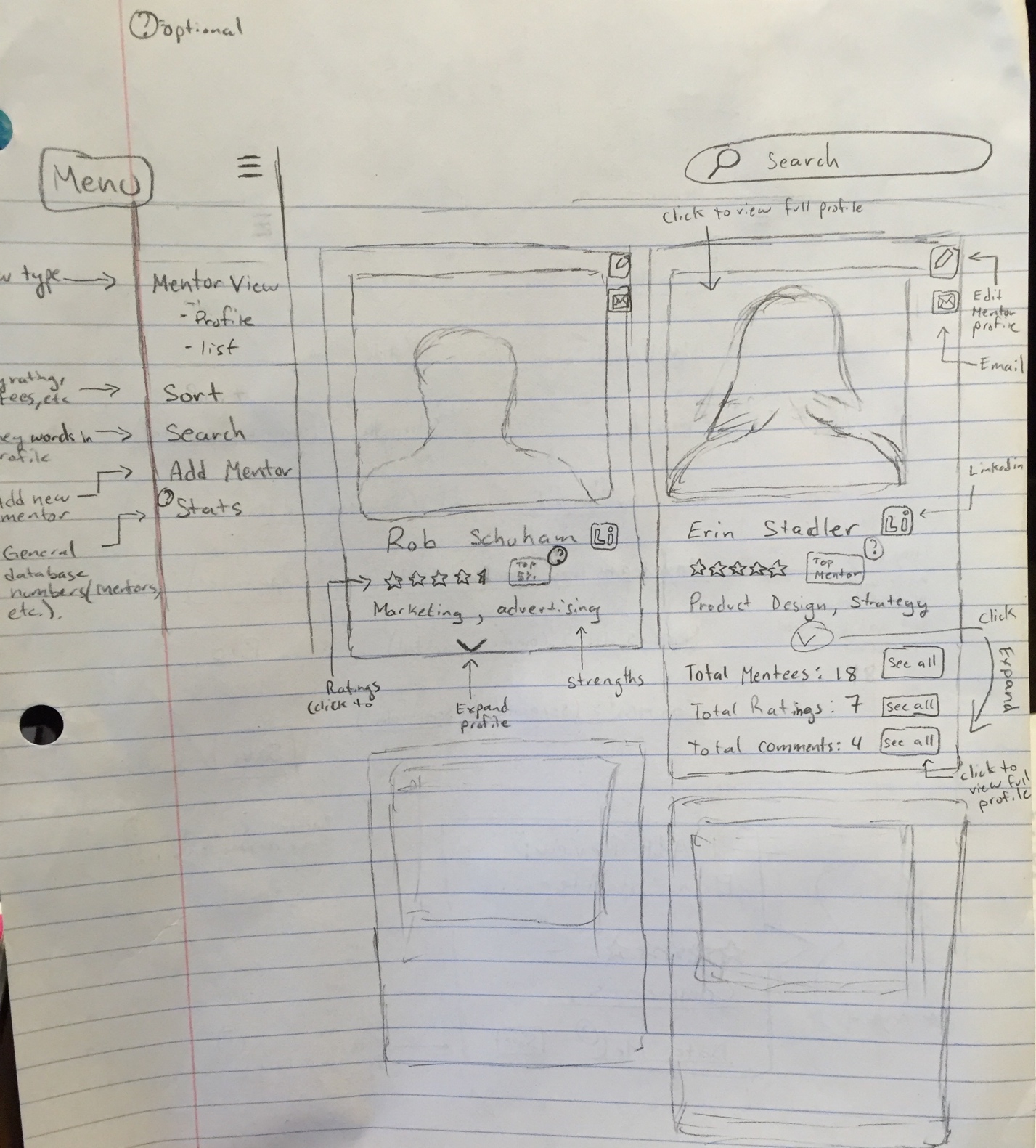
Database UML Design Diagram

**

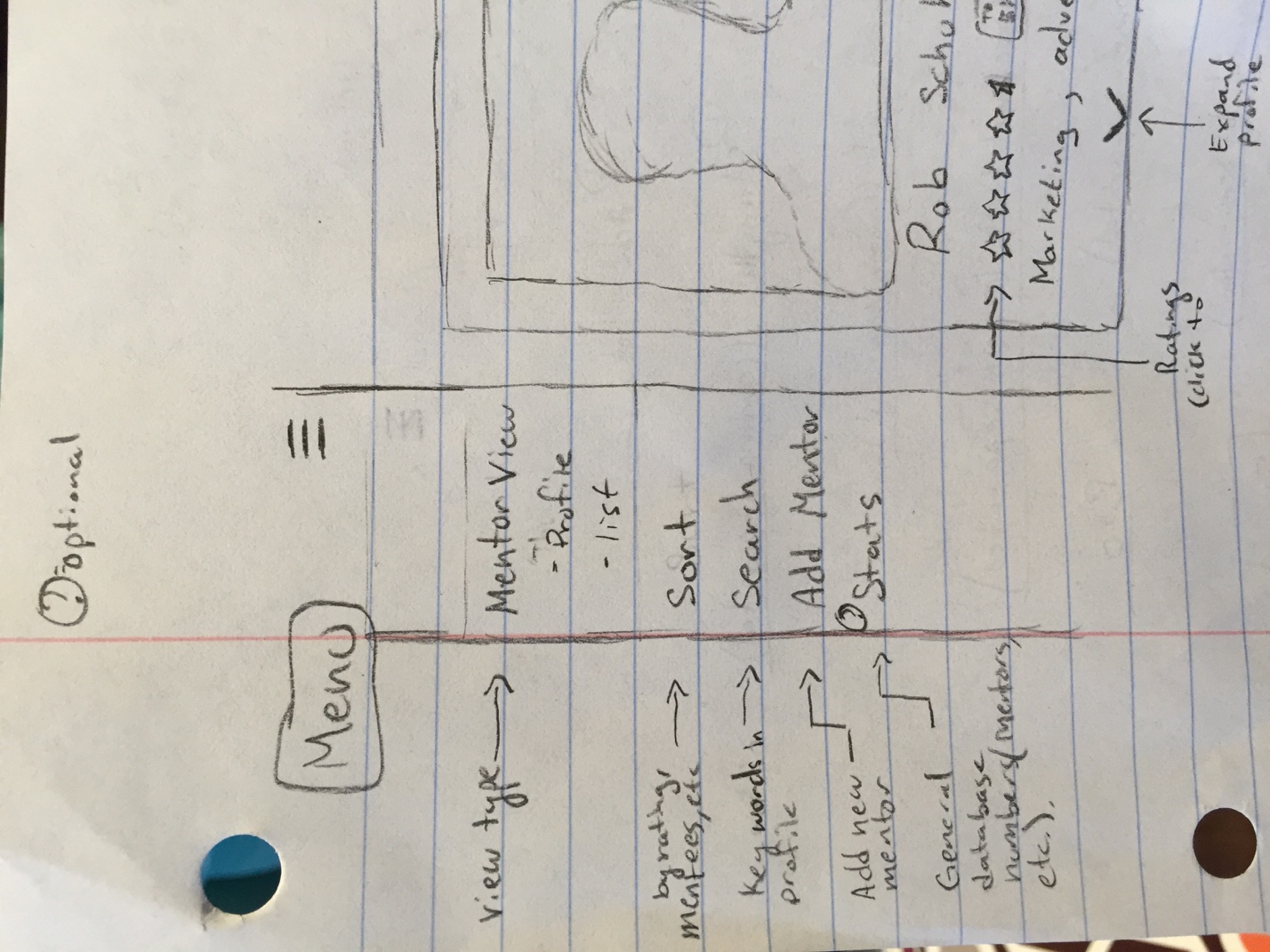
****

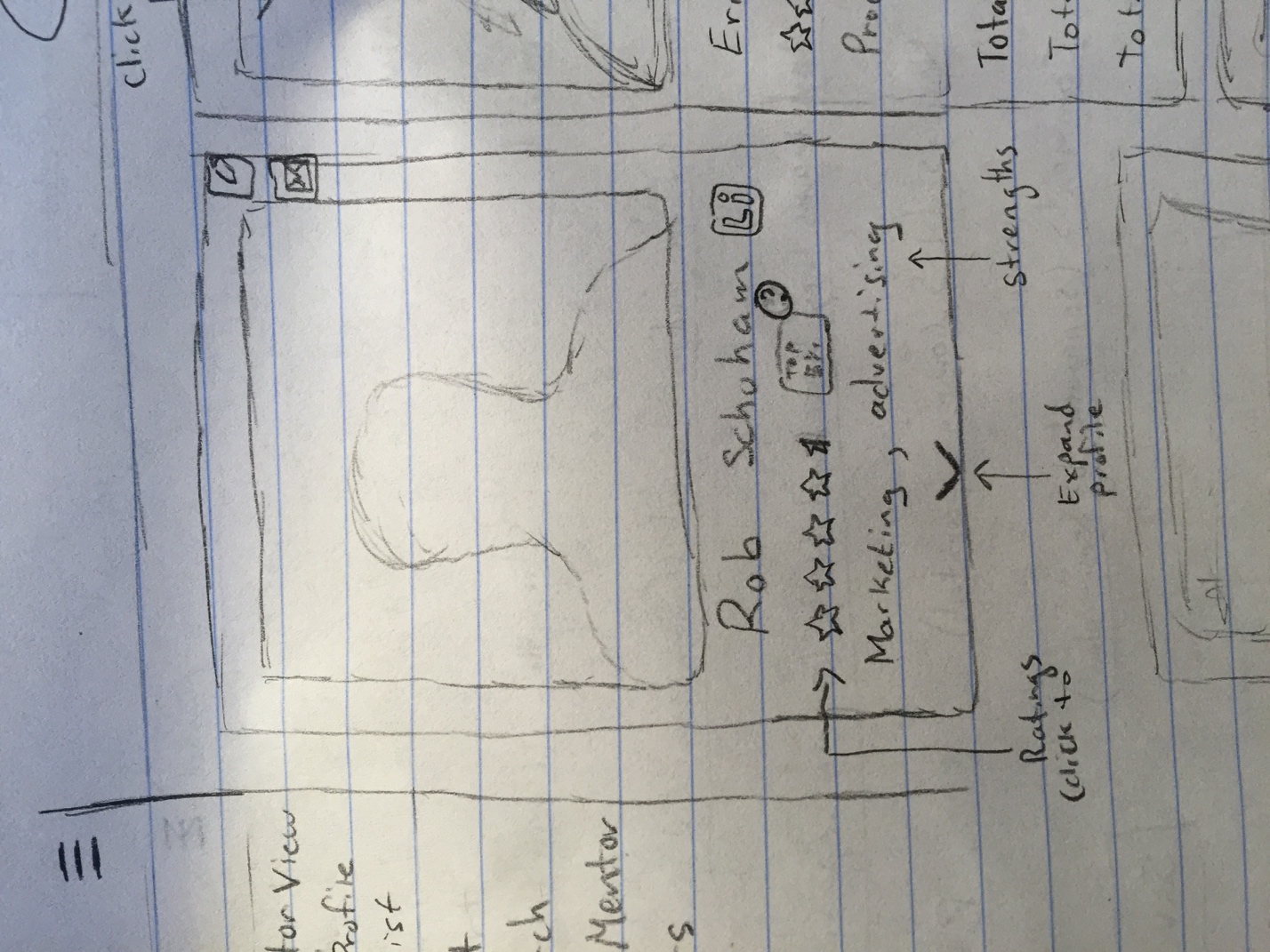
**UI Mockups:**

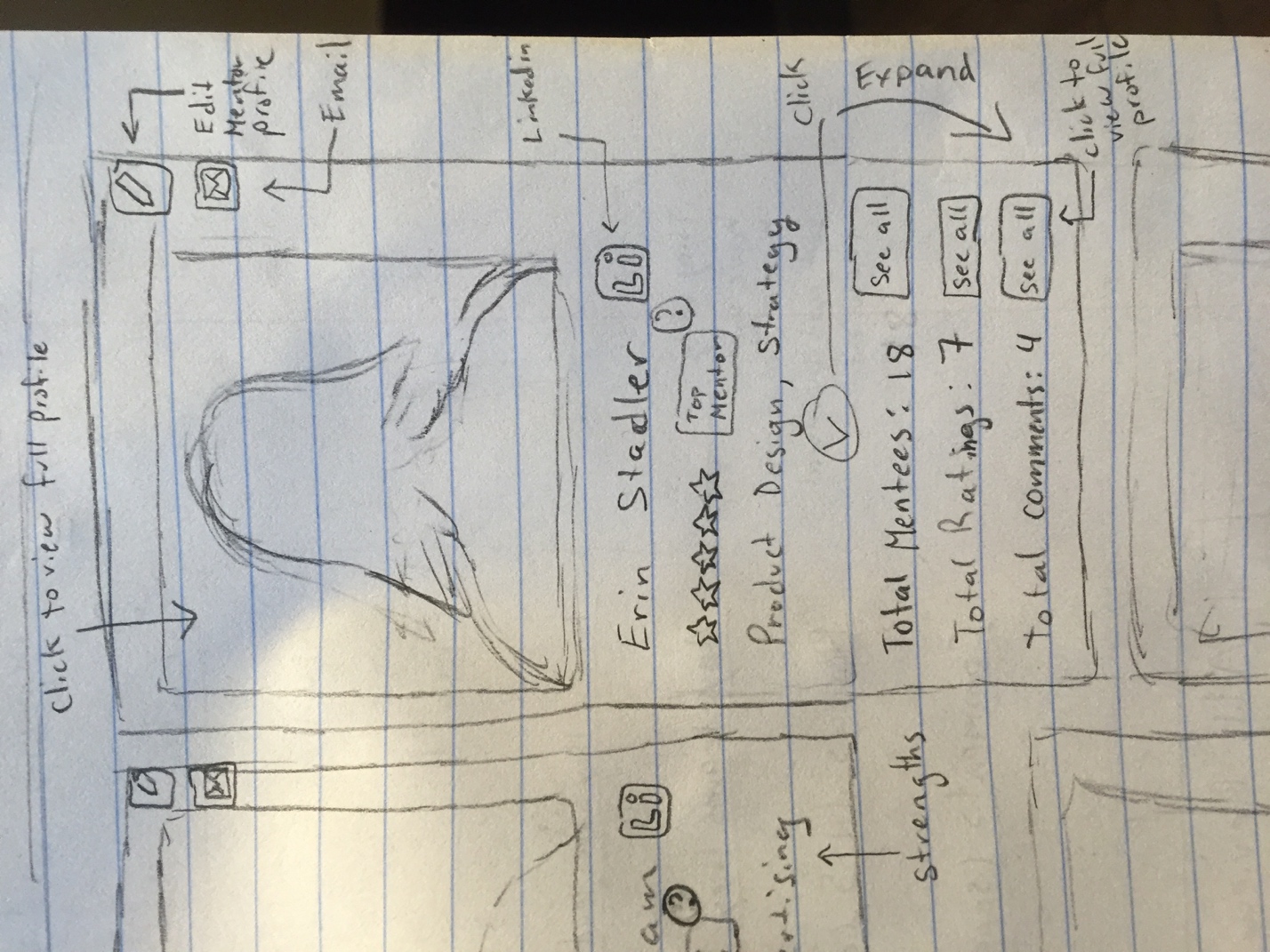
Main page (Menu, and mentor list)

****

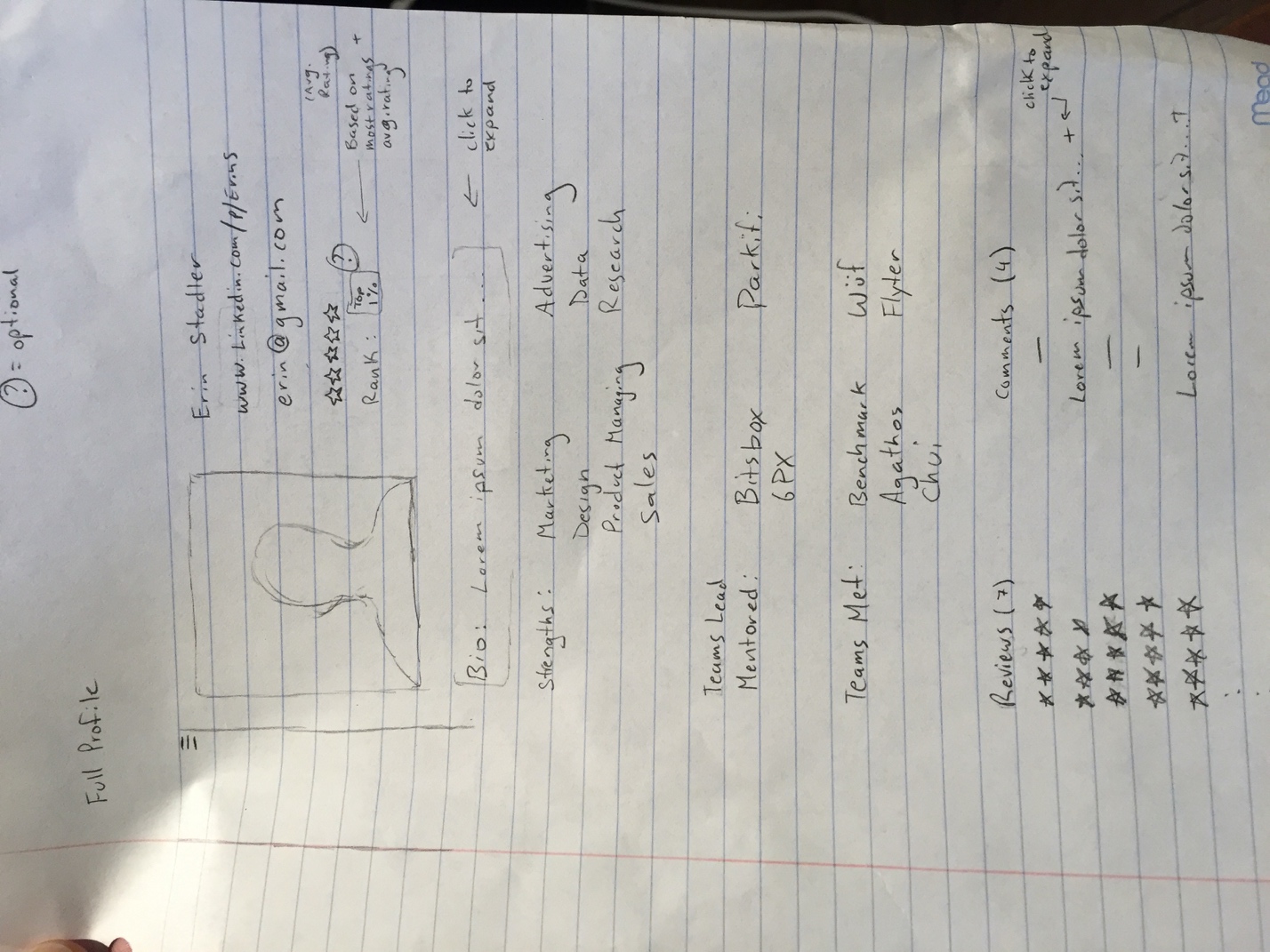
Menu (zoomed in)

****

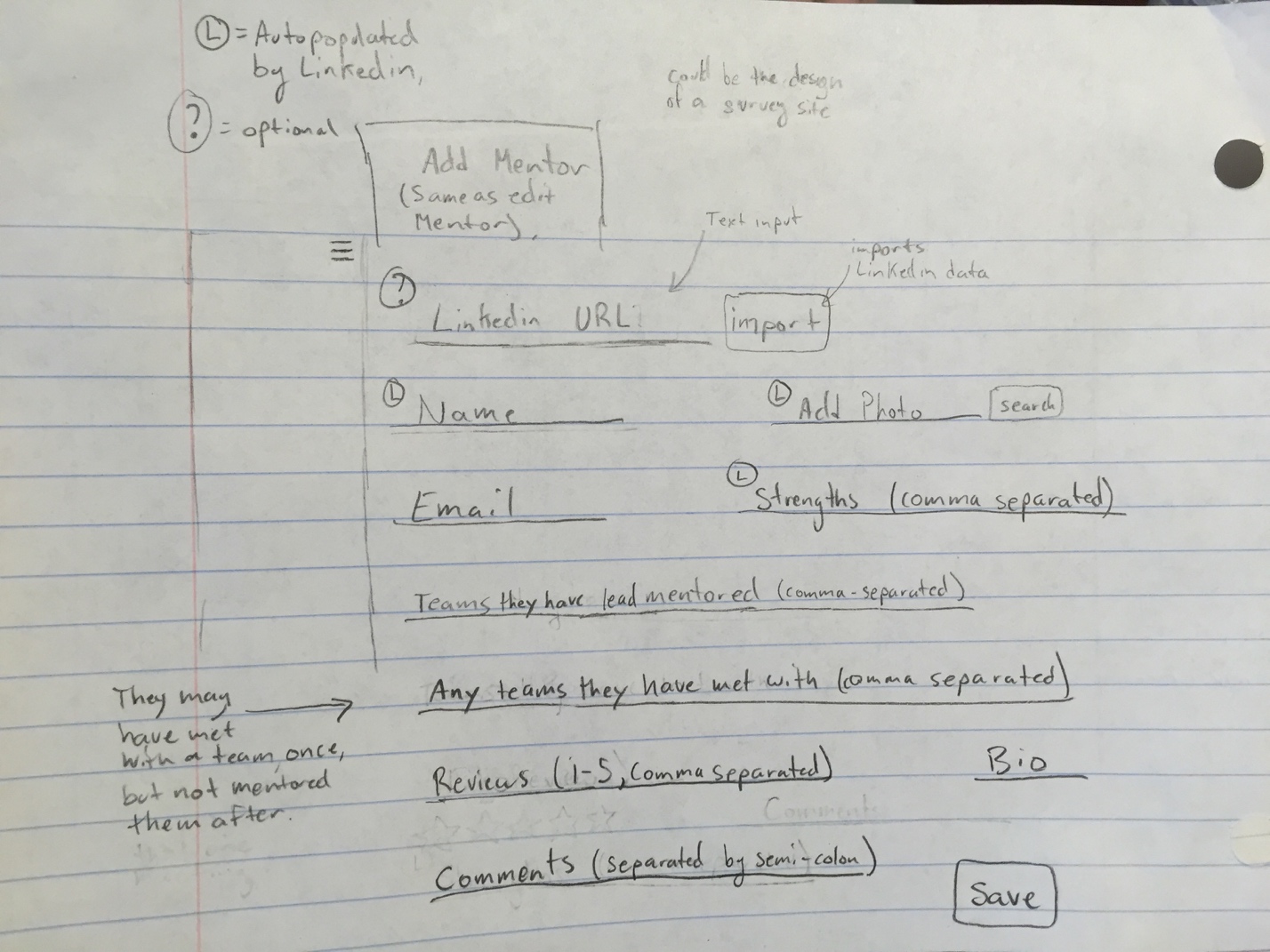
Mentor list View (Zoomed in to one mentor) ****

Some mentor details expanded in mentor list view****

Full Profile View

****

Edit Profile

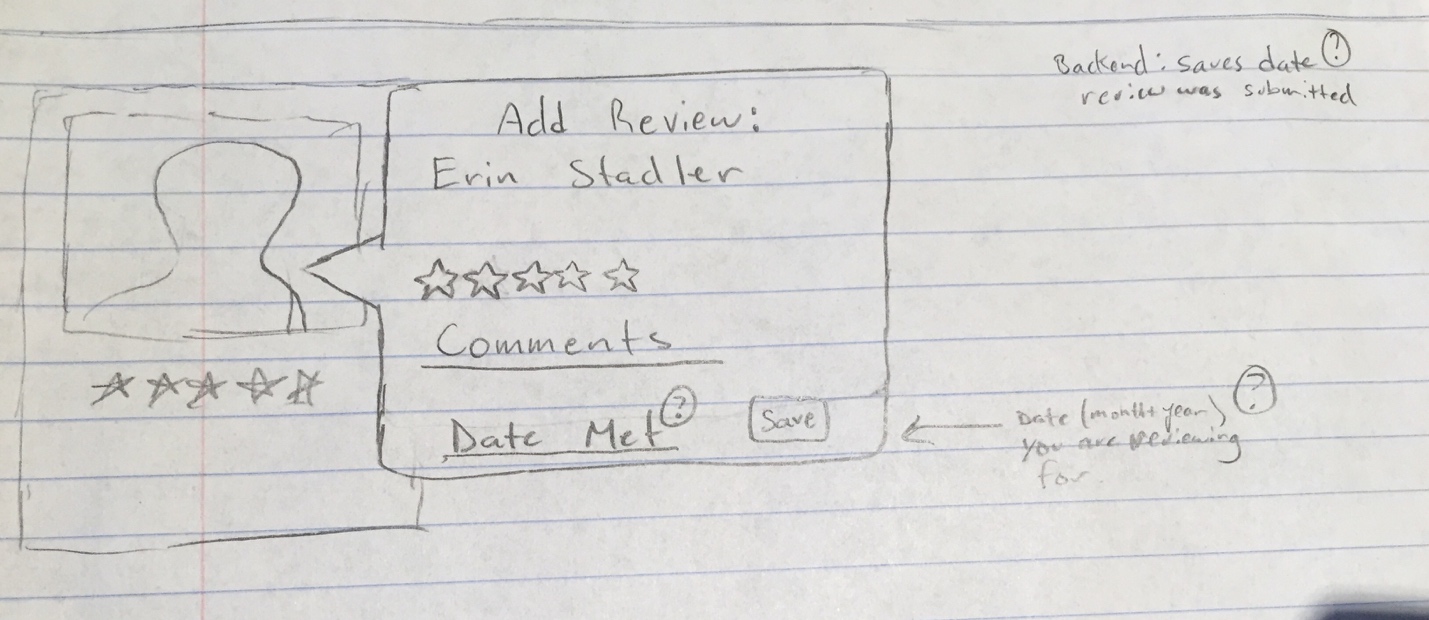
****

Sort function

****

Optional function:

hover over a profile to add feedback

****

**User Interactions:**

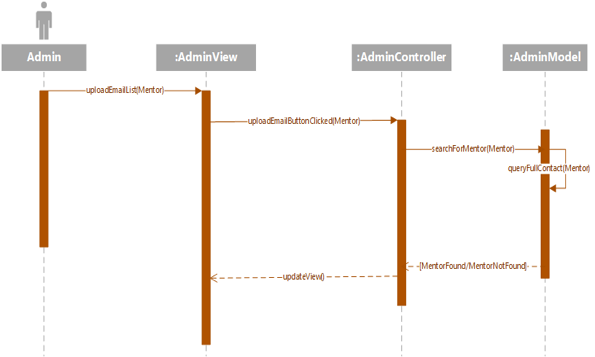
**Sequence Diagram**

**Requirement ID: FR-001**

**Use Case ID: FR-001 (Store a mentor’s LinkedIn profile)**

**Use Case Description: Store a mentor’s LinkedIn profile in the database**

**Person Who Implemented: Michael Bernabei**

****

**Class Diagram:**

