CS 3354 Software Engineering
Final Project Deliverable 1

Carrot Critic

"*Carrot Over Stick"*

Anton Joseph
Navya Monga
Steven Nall
Nathan Price
Dalton Russell
Hassana Saade
Brad Stover

1. [5 POINTS] Please attach here the Final Project draft description (that contains the instructor feedback). It is ok to include a picture of the original document. Address the feedback provided for your proposal by listing what you are did to comply with the proposed changes/requests for additions to your project.

Title: V.A.R. (verified and anonymous reviews)

Group members:

1. Brad Stover
2. Navya Monga
3. Hassana Saade
4. Dalton Russell
5. Nathan Price
6. Steven Nall
7. Anton Joseph

What we will be doing:

We are working together on the design for a mobile app that lets a user anonymously review nearby places of interest, such as restaurants, stores, and local businesses. The app will use the user's location to see the nearby places they can review, or look at reviews left by other users. User reviews will include a star count from 1 to 5 (worst to best), questions the reviewer answered that our app asked about the location (quality of food, customer service, cleanliness, etc.), and extra comments the reviewer can leave. None of the reviews left by users will have any personal information, which is the motivation for our anonymous reviewing app. ✓

Motivation:

We want to provide a counterforce to manipulative individuals who are paid to promote businesses or products through the practice of user deception. Moreover, we wish to prevent shameless and fraudulent self-promotion due to fraudulent reviews. We also wish to protect reviewers from potential backlash or shame as a result of their feedback and to give consumers a review source that is more trustworthy than Yelp or Google reviews.

Where our design would be used:

Our design would be used by customers and critics of local businesses who want to give or review ratings and information about local businesses through an easy to use and trustworthy mobile application. Furthermore, the data gathered by our site could be marketed to other information-based companies to be used in providing ratings in many other applications.

Tasks:

Brad – Group coordinator
Navya – Cloud/Box
Hassana – Github coordinator
Dalton – Process modeling
Nathan – Diagrams
Steven – Project comparison
Anton – .ppt Creation

} Four.

There are many similar sw already implemented. So, pls make sure to differentiate your design so that it becomes uniquely different. Also, include a comparison w/ such similar sw in your final report.

Addressing the feedback:

Unlike most current review services, Carrot Critic will be anonymous for users. This feature will help them avoid potential pitfalls that could stem from using this type of application, such as backlash, embarrassment, and bribery. Furthermore, our reviews will be verified using two-factor authentication, which will help keep out fraudulent reviews and make our service more reliable and trustworthy than its competitors.

Each review a user submits will earn them 1 (one) point. There will be four tiers for user levels based on point spread. Platinum users (those who have submitted 200 reviews or more) will be entitled to the most extensive benefits. These benefits will come from our networking with local businesses and advertising for them. For example, in exchange for our advertising, perhaps the business will supply us with a certain number of coupons. These will be passed on, with platinum users getting exclusive priority access. Users in lower tier categories will also have chances to obtain these discounts, but it is more likely that they'll have to take a survey first.

In summary, anonymity and rewards are how we plan to set our application apart from Yelp and Google Reviews, the 2019 juggernauts in the review applications arena.

1. [10 POINTS] Setting up a GitHub repository:
1.1. Each team member should create a GitHub account if you don't already have one.
1.2. Create a GitHub repository named 3354-teamName. (whatever your team name will be).
1.3. Add all team members, and the TA as collaborators. Our TA will post his GitHub account info once he sets it up.
1.4. Make the first commit to the repository (i.e., a README file with [team name] as its content).
1.5. Make another commit including a pdf/txt/doc file named "project_scope". If you choose a predefined topic (one of the 4 topics described in the "Project Topic Ideas" section of this document), the contents of the file should be identical to the corresponding project in this section. If you choose other topics, the contents should follow a similar structure.
1.6. Keep all your project related files in your repository as we will check them. Include the URL of your team project repository into your project deliverable 1 report.
Important Note:

- Tasks 1.3 - 1.5 should be performed by different team members. We will check the commit history for these activities.
- Do not include credentials (e.g., UTD ID) in the repository.
- Only commits performed before the deadline will be considered. Do not forget to push your changes after you have done the work!

- ✓ 1.3 - Completed by Hassana
- ✓ 1.4 - Completed by Brad and Navya
- ✓ 1.5 - Completed by Steven and Navya

Team Project Repository URL:

https://github.com/hassanasaade/SE3354-CARROTCRITIC.git

2. **[5 POINTS] Delegation of tasks: Who is doing what**

Group Meeting Coordinator – Navya

GitHub Coordinator – Hassana

Software Model - Nathan

Software Requirements – Navya, Steven and Brad

Use Case Diagrams – Anton and Brad

Sequence Diagrams - Hassana and Dalton

Class Diagrams – Anton and Navya

Software Architecture – Steven

3. **[5 POINTS] Which software process model is employed in the project and why. (Ch 2)**

Software model based on group consensus: Incremental model.

We chose the incremental model because our product needs to be flexible. We will have constantly changing user requirements as we gather feedback and make updates over time. It is very difficult to prototype a review app, and we don't have significant need for risk management due to the relative simplicity of our system. We therefore felt no need to utilize the prototyping or spiral models. The incremental model will allow us to create the product effectively and without unnecessary overhead.

4. **[15 POINTS] Software Requirements including**

4.a.) [5 POINTS] Functional requirements. To simplify your design, please keep your functional requirements in the range minimum 5 (five) to maximum 7 (seven). (Ch 4)

Functional Requirements:

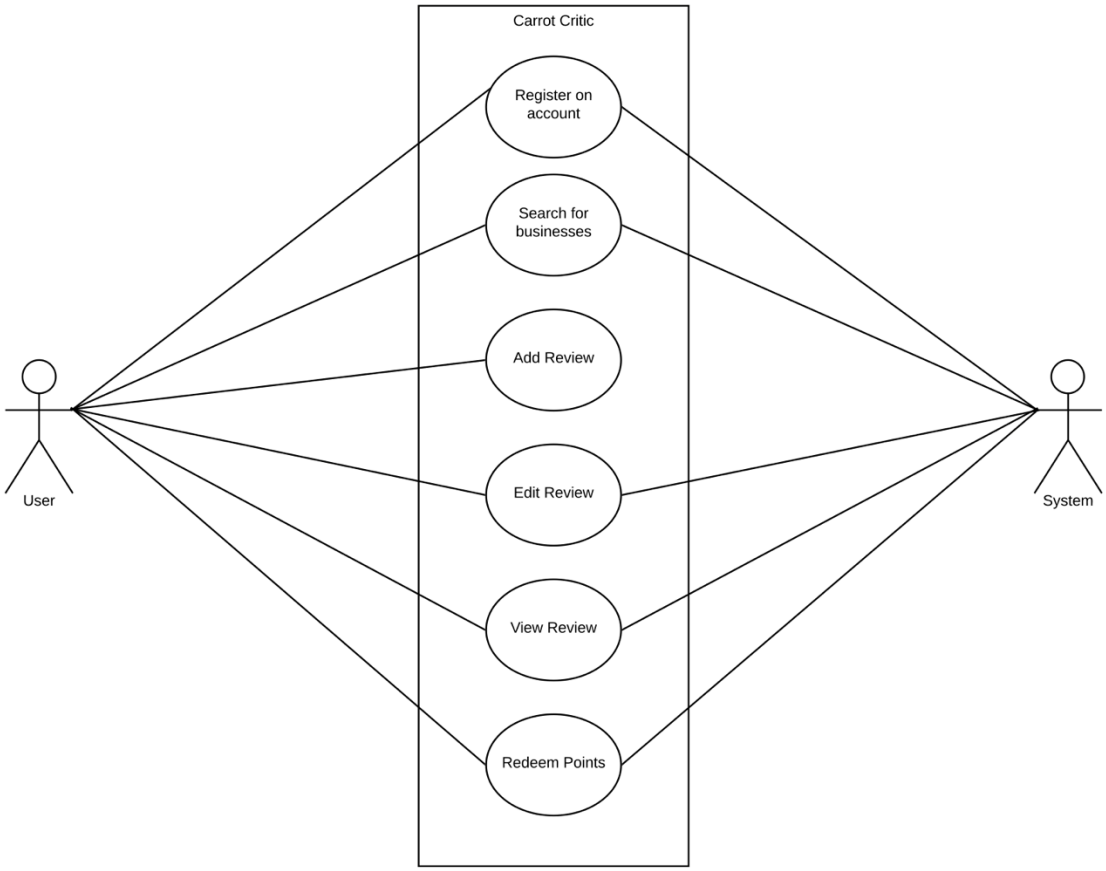
1. The user shall not be able to register more than one account.
2. The user shall be able to search for nearby businesses based on type, cost, distance, and so forth.
3. The user shall be able to add a review if desired.
4. The user shall have the ability to search through reviews for given businesses.
5. The user shall be able to edit the content of their previous reviews.
6. The user shall be able to redeem points for rewards.

4.b.) [10 POINTS] Non-functional requirements (use **all** non-functional requirement types listed in Figure 4.3 - Ch 4)

Nonfunctional Requirements:

1. Product Requirements:
    a. Usability Requirement: Accessibility aids shall be heavily utilized. These include but are not limited to: a narrator to read text on user screen, magnifier capability to increase size of text and button (as well as decrease if necessary), color inversion, optional simplified UI (with larger buttons and omitted complexities), visual/interactive tutorials, and more.
    b. Efficiency Requirements:
        i. Performance Requirements: The system shall display search results to user within 5 seconds.
        ii. Space Requirements: The system shall send weekly push notifications to the user and ask them if they would like to update the local data on their device. The local database itself shall be custom-tailored to their search history.
    c. Dependability Requirement: The system shall be only down for maintenance at a given weekly time interval (Tuesdays from 3 to 4am CST.) During this period, users cannot create new reviews but may still access the database.
    d. Security Requirement: The system shall protect the integrity of user data (location, IP address, etc.) by storing user data on user devices.
2. Organizational requirements:
    a. Environmental Requirements: The product shall be compatible with Android (6.0 and higher) and iOS (v10 and higher) devices.
    b. Operational Requirements: The product shall explicitly request access to location services as well as contact details for verification purposes.
    c. Development Requirements: The product shall be organized according to the principles of Model-View-Controller. The inner workings of the application shall be entirely separated from the UI development.
3. External Requirements:
    a. Regulatory Requirements: The application shall operate within the constraints set by the respective regulatory authority in each country in which it is used.
    b. Ethical Requirements: Data shall not be influenced by businesses; rather, data shall be solely determined by end user input.
    c. Legislative Requirements:
        i. Accounting Requirements: Privacy protection technicalities shall be explicitly written into the user terms & conditions agreement.
        ii. Safety/Security Requirements: Account details shall be accessible only to the account holder.

5. **[15 POINTS] Use case diagram – Provide a use case diagram (similar to Figure 5.5) for your project. Please note than there can be more than one use case diagrams as your project might be very comprehensive. (Ch 5 and Ch 7)**



Use Case Diagram Tabulations:

1) **Register Account**

Function: Allow prospective user to register an account.

Description: Verifies email address. Verifies phone number via unique six-digit code. Adds user to database. Creates user profile.

Inputs: Email address from user. User's cell phone number. Six-digit code sent to user.

Source: User database.

Outputs: User profile in database.

Destination: Main control loop.

Action: Call getEmail() to obtain user input. Verify it with setEmail(). If verification is successful, indicate this to user. Otherwise, display error message. Obtain phone number using getPhoneNumber(). Verify it with setPhoneNumber() and getUniqueId(). If the given phone number is successfully authenticated, display success; otherwise, display error message. If both email and phone number pass verification, register user's account in database and allow them to customize their new profile.

Requirements: An email and phone number that pass validation tests and have not yet been used.

Pre-condition: The user has not already signed up for an account.

Post-condition: The user's profile is created in the database; it is now open to them for customization.

Side effects: None.

2) **Search For Business**

Function: Allow user to search for businesses.

Description: Checks user inquiry against list of businesses in database. Displays error message if no relative or absolute match is found. Shows list of matches otherwise.

Inputs: User text related to search input.

Source: Database

Outputs: List of absolute and relative search matches in terms of businesses matching the user inquiry.

Destination: Main control loop.

Action: Allow user to type in a simple search query relating to the business name they wish to look up. Check query against database for similar terms or terms that match the user query exactly. Use searchBusiness() function to achieve this. If search is successful, return relevant results using returnList(). Otherwise, display error message.

Requirement: A database that has kept up-to-date with local business names and information.

Pre-condition: The user's input has been properly formatted (proper spacing, grammar, and spelling).

Post-condition: A list of results is displayed to the user with a preview of details about each match.

Side effects: None

3) **Add Review**

Function: Allow user to add a review.

Description: Allows user, in a finite number of characters, to write a review for a business. Assures that their credentials are valid using 2FA login.

Inputs: User credentials, data from 2FA login, text from user.

Source: Database of reviews.

Outputs: Read-only portion of review database to be accessed by the user.

Destination: Main control loop.

Action: Call searchBusiness() function. If the search is successful, allow the user to select the desired business. Ensure that they are logged in; if not, validate credentials before doing anything else. Display error message and do not let the user proceed if their credentials are invalid. Call the addReview() function to accept user text input, converting it into a live review. Display the now-live review to the user using the viewReview() function.

Requirement: The user has not reviewed the business before. User credentials are valid.

Pre-condition: The searchBusiness() function returns at least one valid result.

Post-condition: The user's review is published and displayed to them as confirmation.

Side effects: None.

4) **Edit Review**

Function: Allow user to edit their own review.

Description: Allows user to revise a previous review that they authored. Assures that their credentials are valid using 2FA login. Displays revised review to user for clarification purposes.

Inputs: User credentials, data from 2FA login, text from user.

Source: Database of reviews.

Outputs: Read-only portion of review database to be accessed by the user.

Destination: Main control loop.

Action: Call searchBusiness() function. If the search is successful, allow the user to select the desired business. Ensure that they are logged in; if not, validate credentials before doing anything else. Display error message and do not let the user proceed if their credentials are invalid. Call the editReview() function to allow text editing for previous review. Display the now-live revised review to the user using the viewReview() function.

Requirement: The user has reviewed the business before. User credentials are valid.

Pre-condition: The searchBusiness() function returns at least one valid result.

Post-condition: The user's revised review is published and displayed to them as confirmation.

Side effects: None.

5) **View Review**

Function: Allow user to view a review.

Description: Allows user to view all reviews (if applicable) for a business. Assures that their credentials are valid using 2FA login.

Inputs: User credentials, data from 2FA login, search query from user.

Source: Database of reviews.

Outputs: Local database to privately store user query for better search customization in the future.

Destination: Main control loop.

Action: Call searchBusiness() function. If the search is successful, allow the user to select the desired business. Ensure that they are logged in; if not, validate credentials before doing anything else. Display error message and do not let the user proceed if their credentials are invalid. Call the getReview() function to pull the desired review from the database. Call viewReview() to fetch that review for the user and display it in full detail.

Requirement: User credentials are valid.

Pre-condition: The searchBusiness() function returns at least one valid result.

Post-condition: The desired review is displayed to the user.

Side effects: None.

6) **Redeem Rewards**

Function: Allow user to redeem their points for reward(s).

Description: Allows user to fetch the number of points they currently have, as well as their current tier level. If applicable, allows the user to redeem their points for a given reward.

Inputs: User inquiry.

Source: Database of user point records.

Outputs: Display of statistics related to point data and eligible rewards.

Destination: Main control loop.

Action: Call getPoints() function. If the user has points stored up, return detailed statistics using returnPoints(). These shall include their current tier level, point standing, and eligible rewards. If they are eligible for a certain reward and wish to redeem points for it, allow them to do so using the redeemPoints() function.

Requirement: The user has points stored up and is currently at bronze, silver, gold, or platinum level (based on the number of reviews they have submitted to us).

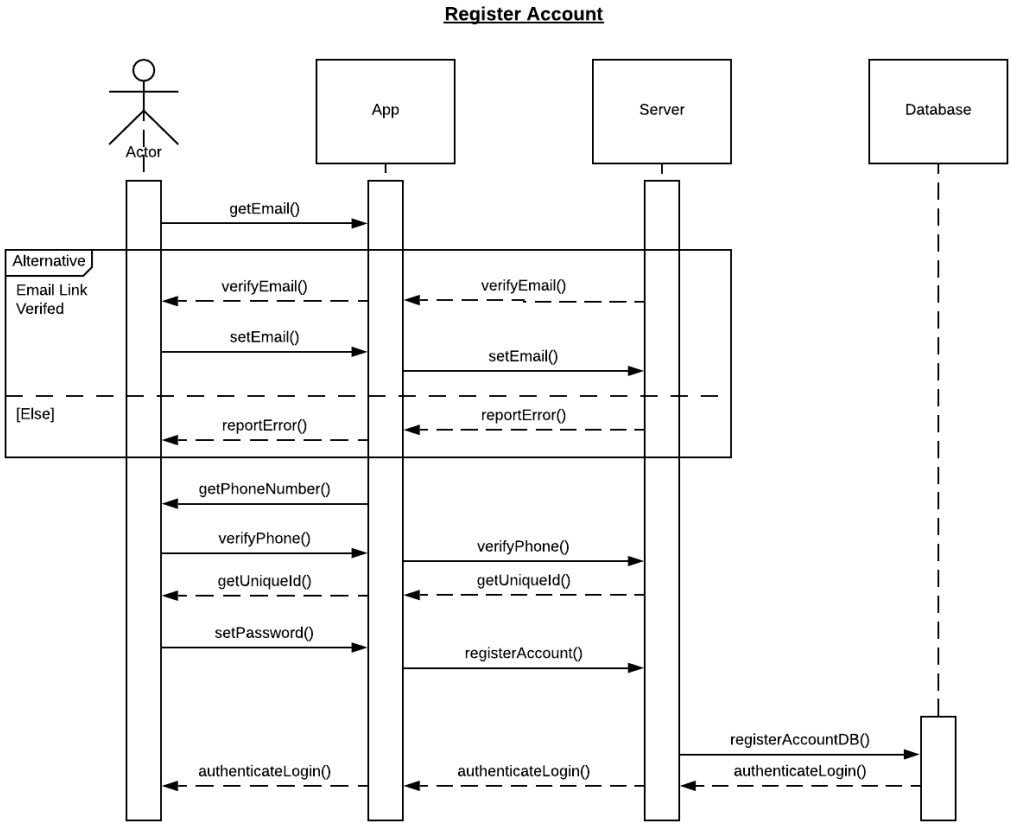
Pre-condition: The user has points stored up from previous reviews written.

Post-condition: Eligible rewards are, at a bare minimum, displayed. They are redeemed if the user so desires.

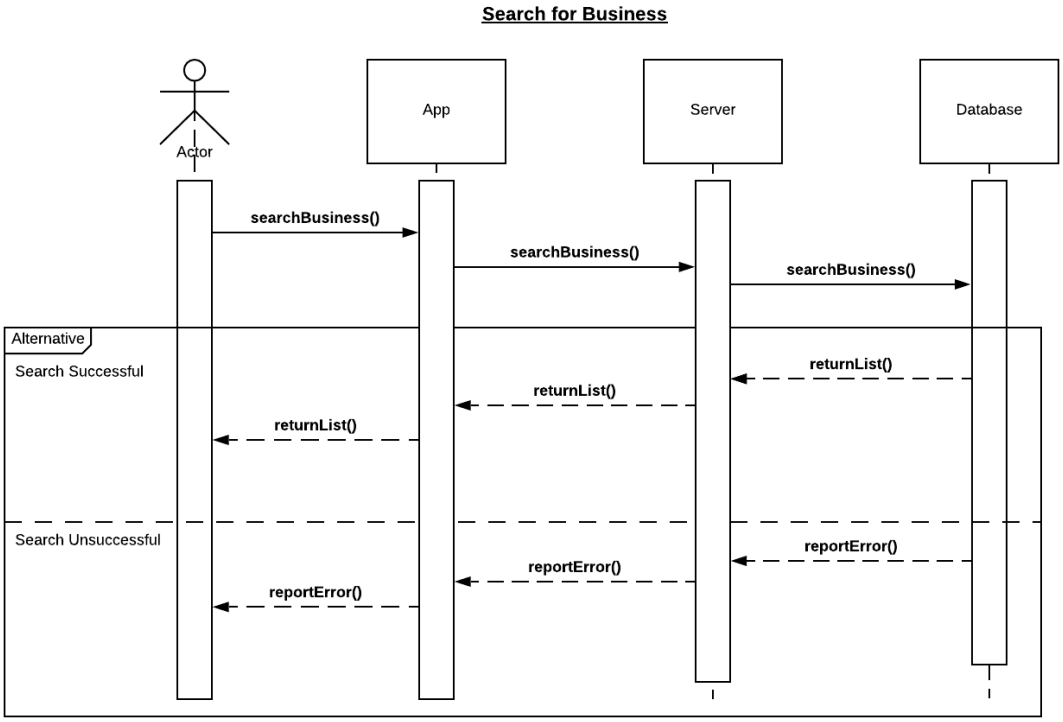
Side effects: None.

6. **[15 POINTS] Sequence diagram – Provide sequence diagrams (similar to Figure 5.6 and Figure 5.7) for each use case of your project. Please note that there should be an individual sequence diagram for each use case of your project. (Ch 5 and Ch 7)**

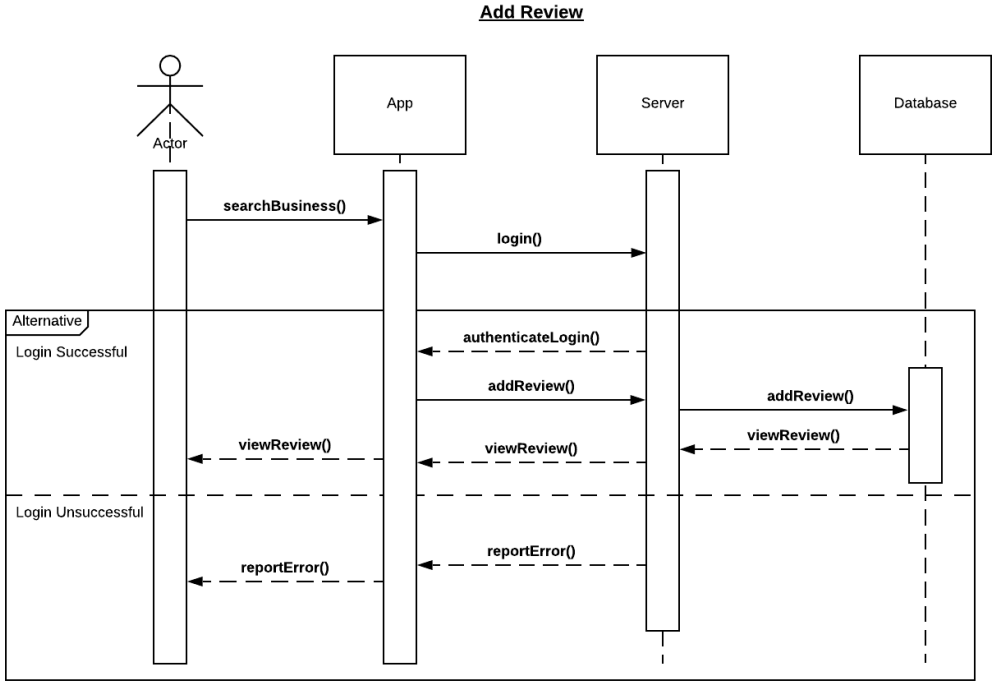
1. **<u>Register Account</u>**



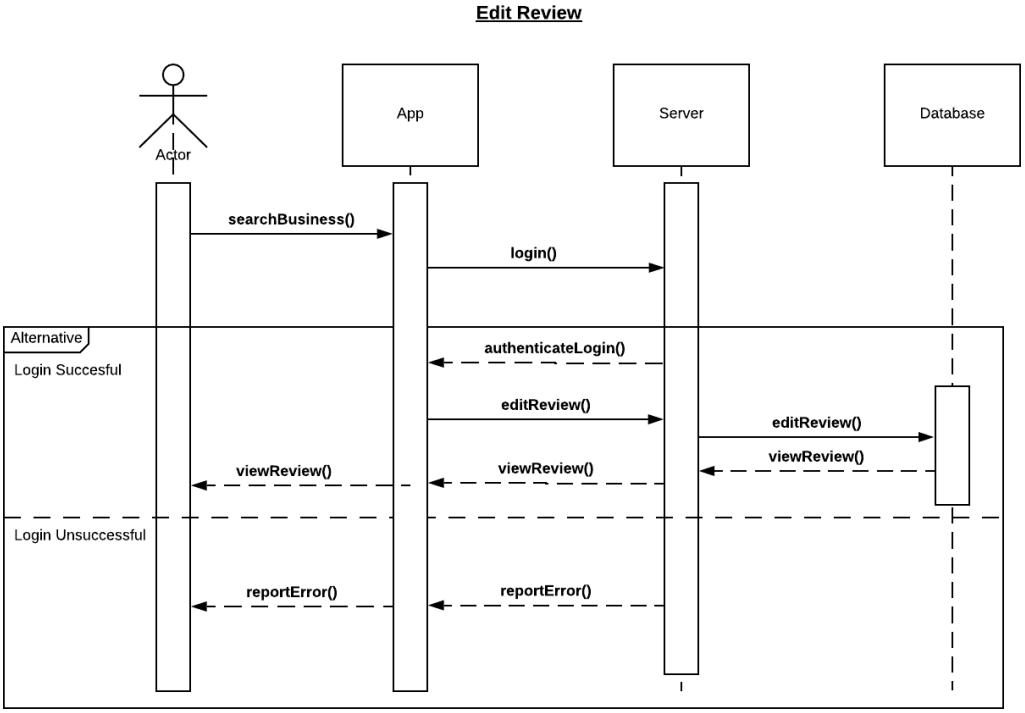
2. Search For Businesses



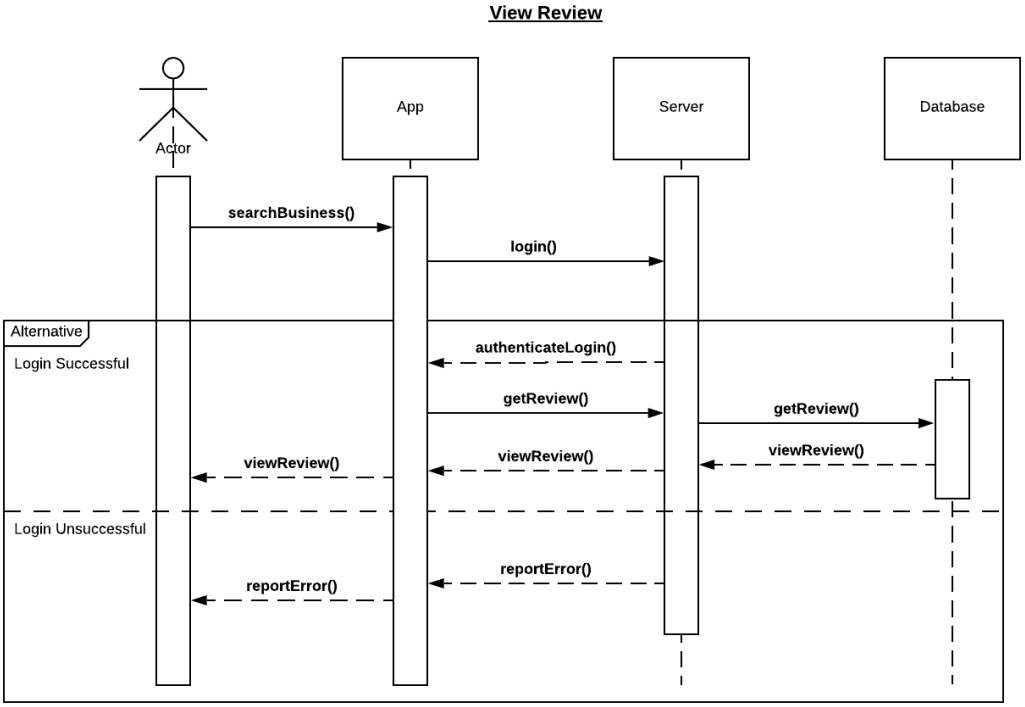
3. Add Review



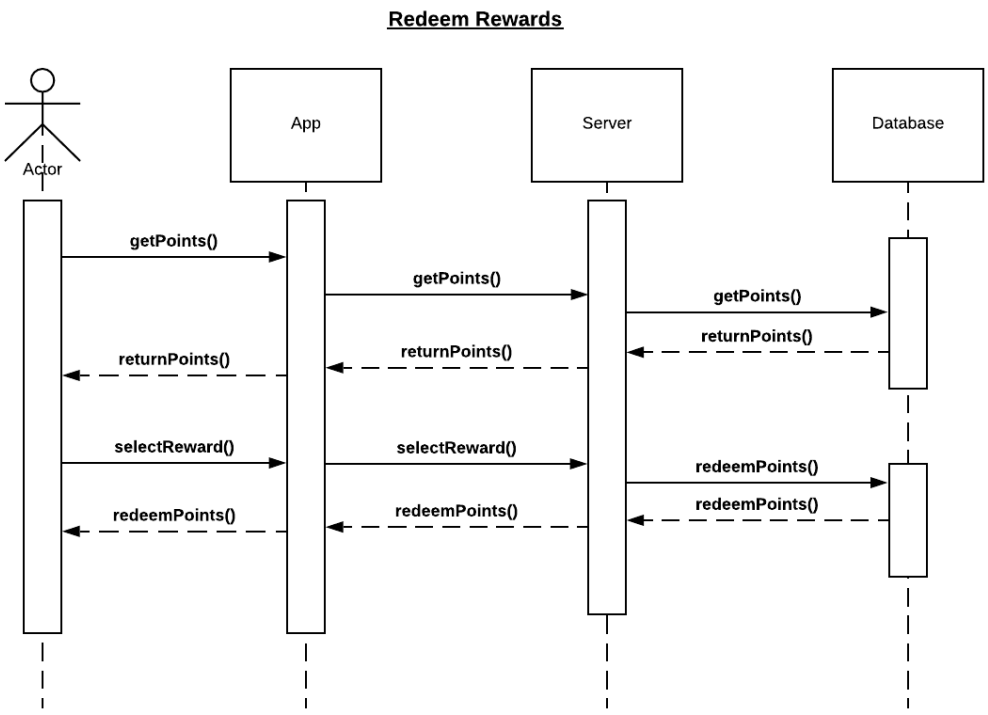
4. Edit Review



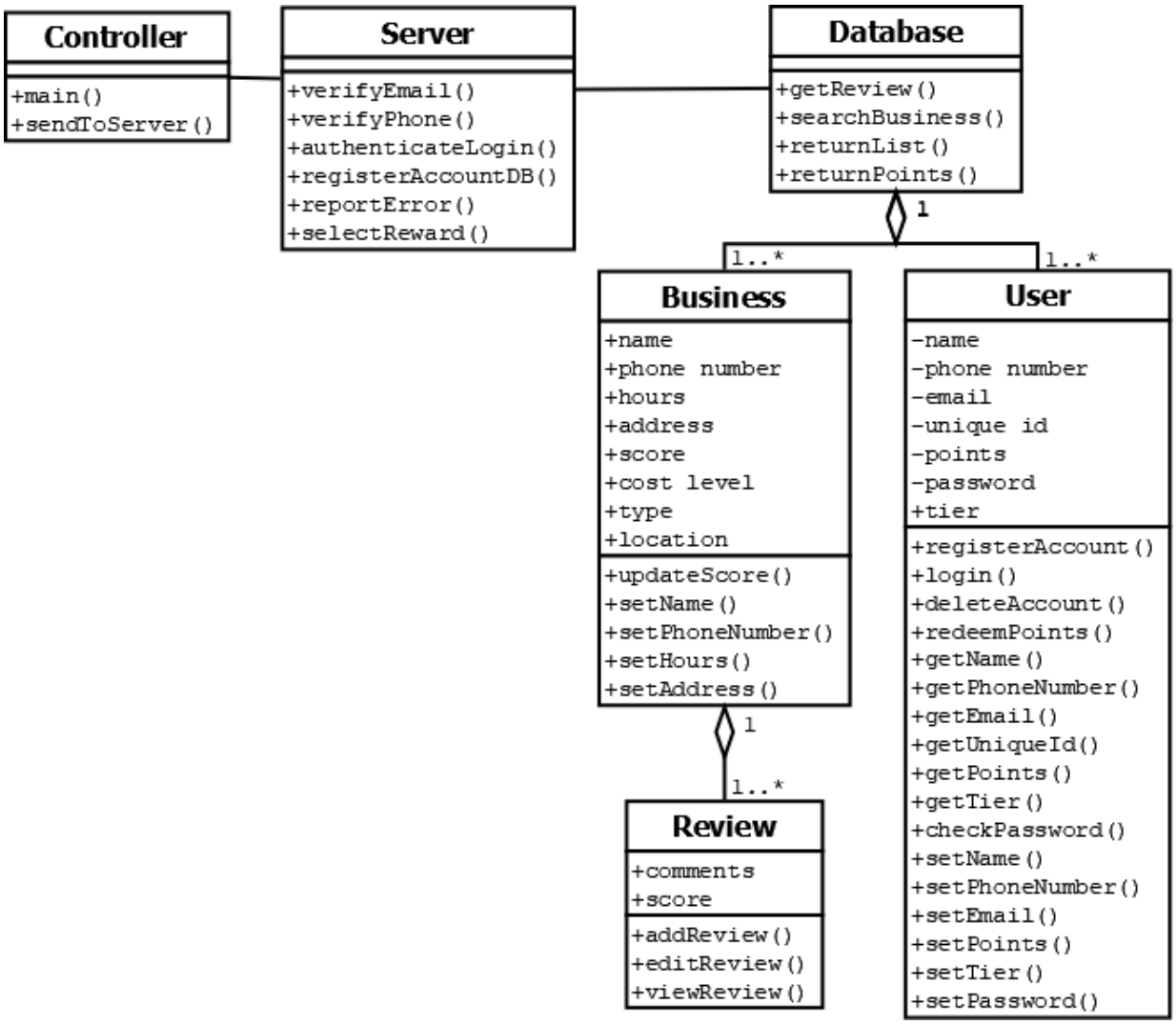
5. View Review



6. Redeem Rewards



7. **[15 POINTS] Class diagram – Provide a class diagram (similar to Figure 5.9) of your project. The class diagram should be unique (only one) and should include all classes of your project. Please make sure to include cardinalities, and relationship types (such as generalization and aggregation) between classes in your class diagram. Also make sure that each class has class name, attributes, and methods named (Ch 5).**



8. **[15 POINTS] Architectural design – Provide an architectural design of your project. Based on the characteristics of your project, choose and apply only one appropriate architectural pattern from the following list: (Ch 6 section 6.3)**
    a. **Model-View-Controller (MVC) pattern (similar to Figure 6.6)**
    b. **Layered architecture pattern (similar to Figure 6.9)**
    c. **Repository architecture pattern (similar to Figure 6.11)**
    d. **Client-server architecture pattern (similar to Figure 6.13)**
    e. **Pipe and filter architecture pattern (similar to Figure 6.15)**

The Software Architecture Model we chose was the Model-View-Controller (MVC), as exemplified below: