**Technical Design Document – Patchwork Love**

**By:** Alex Bates,Junith Seals

**Executive Summary**

Our service is a dating website. It is themed around stuffed animals, and all the profiles are stuffed animals, that way development is fun. The problem we solve is loneliness and the chronic lack of connection and coziness that faces society. Users will be experiencing the formula pioneered by Tinder of swiping through profiles until you get a match, then the conversation begins.

Graphical user interface

Description automatically generated with medium confidence**Services Diagram**

**Services API Documentation**

**Profile Service**

* /profile
  + Action: Creates a Dating Profile
  + Verb: POST
  + Code: 201
  + Request Body:
    - Name: “string”
    - Bio: “string”
    - AgePrefs: “int[] length of 2”
    - Pictures: “string[]”
    - Interests: “string[]”
    - LookingFor: “string”
  + Response Body:
    - Name: “string”
    - Bio: “string”
    - AgePrefs: “int[] length of 2”
    - Pictures: “string[]”
    - Interests: “string[]”
    - LookingFor: “string”
    - Matches: “MATCH OBJECT[]”
    - OutgoingLikes: “LIKE OBJECT[]”
    - Dislikes: “string[]”
    - doNotifs: “bool”
    - isHidden: “bool”
* /profile/{profileID or the string “me”}
  + Action: Update a Dating Profile
  + Verb: PATCH
  + Code: 200
  + Request Body:
    - Name: “string”
    - Bio: “string”
    - AgePrefs: “int[] length of 2”
    - Pictures: “string[]”
    - Interests: “string[]”
    - LookingFor: “string”
    - Matches: “MATCH OBJECT[]”
    - OutgoingLikes: “LIKE OBJECT[]”
    - Dislikes: “string[]”
    - doNotifs: “bool”
    - isHidden: “bool”
  + Response Body:
    - Name: “string”
    - Bio: “string”
    - AgePrefs: “int[] length of 2”
    - Pictures: “string[]”
    - Interests: “string[]”
    - LookingFor: “string”
    - Matches: “MATCH OBJECT[]”
    - OutgoingLikes: “LIKE OBJECT[]”
    - Dislikes: “string[]”
    - doNotifs: “bool”
    - isHidden: “bool”
* /profile/delete/{profileID}
  + Action: Delete Profile
  + Verb: DELETE
  + Code: 204
  + Request Body:
    - None
  + Response Body:
    - None
* /like/{profileID}
  + Action: Send a like
  + Verb: POST
  + Code: 202, 200
  + Request Body:
    - None
  + Response Body:
    - isMatch: “bool”
* /dislike/{profileID}
  + Action: Dislike a Profile
  + Verb: POST
  + Code: 204
  + Request Body:
    - None
  + Response Body:
    - None
* /profile/get/{count}
  + Action: Get batch of profiles
  + Verb: GET
  + Code: 200
  + Request Body:
    - None
  + Response Body:
    - Profiles: “PROFILE OBJECT[]”
* /profile/get\_matches
  + Action: Get all matches
  + Verb: GET
  + Code: 200
  + Request Body:
    - None
  + Response Body:
    - Profiles: “PROFILE OBJECT[]”

**Message Service**

* /send
  + Action: Send a message to a user (updates match object)
  + Verb: POST
  + Code: 201
  + Request Body:
    - Sender: “GUID”
    - Recipient: “GUID
    - Text: “string”
    - timeSent: “string”
  + Response Body:
    - None
* /match
  + Action: Create a match in the database
  + Verb: POST
  + Code: 201
  + Request Body:
    - First: “GUID”
    - firstMsg: “string”
    - Second: “GUID”
    - secondMsg: "string”
  + Response Body:
    - None
* /unmatch/{sender}
  + Action: Remove a match
  + Verb: DELETE
  + Code: 204
  + Request Body:
    - None
  + Response Body:
    - None
* /retrieve/{sender}
  + Action: Receive new messages
  + Verb: GET
  + Code: 200
  + Request Body:
    - None
  + Response Body:
    - Messages: “string[]”

**Auth Service**

* /register
  + Action: Create new account
  + Verb: POST
  + Code: 201
  + Request Body:
    - Email: “string”
    - Password: “string”
  + Response Body:
    - None
* /login
  + Action: Log in to existing account
  + Verb: POST
  + Code: 200
  + Request Body:
    - Email: “string”
    - Password: “string”
  + Response Body:
    - Token: “JWT”
    - Profile:
      * ID: “GUID”
      * Matches: “string[]”
      * Likes: ”string[]”
      * Dislikes: ”string[]”
      * doNotifs: “bool”
      * AgePrefs: “int[]”
      * isHidden: “bool”
      * Pictures: “string[]”
      * Interests: “string[]”
      * LookingFor: “enum”
      * Bio: “string”
* /update/{userID or the string “me”}
  + Action: Change email or password associated with account, “me” means to update the logged in user
  + Verb: PATCH
  + Code: 204
  + Request Body:
    - Email: “string”
    - Password: “string”
  + Response Body
    - None
* /delete/{userID or the string “me”}
  + Action: Delete specified account, “me” means to delete the logged in user
  + Verb: DELETE
  + Code: 204
  + Request Body:
    - None
  + Response Body:
    - None

**Test Cases**

**Profile Tests**

CreateValidProfile

* Endpoint:/profile
* Description: Test that profile creation works

CreateInvalidProfile

* Endpoint:/profile
* Description: Test that you can’t create a profile with missing or invalid information

UpdateLoggedInUserProfile

* Endpoint:/profile/me
* Description: Test that you can update your own profile

UpdateSpecificUserProfile

* Endpoint:/profile/{profileGUID}
* Description: Test that we can update specific profiles

DeleteSpecificUserProfile

* Endpoint:/profile/delete/{profileGUID}
* Description: Test that profiles can be deleted
  + (Deleted accounts are flagged as “deleted”, but they remain in our database)

SendFirstLike

* Endpoint:/like/{profileGUID}
* Description: Test sending a like that is not yet reciprocated

SendSecondLike

* Endpoint:/like/{profileGUID}
* Description: Test sending a like to a profile that has already liked the active user

SendDislike

* Endpoint:/dislike/{profileGUID}
* Description: Test that sending a dislike works

GetBatchOfProfilesFromNewProfile

* Endpoint:/profile/get/{count}/
* Description: Test that profiles are correctly grabbed, and that the order of “most likeness” is proper (Starts with user with highest “likeness” score

GetBatchOfProfilesFromProfileWithLikes

* Endpoint:/profile/get/{count}/
* Description: Test that profiles are correctly grabbed, and that there are no profiles for people you have already liked

GetBatchOfProfilesFromProfileWithDislikes

* Endpoint:/profile/get/{count}/
* Description: Test that profiles are correctly grabbed, and that there are no profiles for people you have already disliked

GetAllMatches

* Endpoint”/profile/get\_matches
* Description: Test that getting matches works, and puts matches with recent messages to the top of the list

**Message Tests**

SendMessage

* Endpoint: /send
* Description: Test that messages can be sent successfully

SendMessageNoMatch

* Endpoint: /send
* Description: Test that a user cannot send a message to a user who has not matched with them

SendMessageUnMatch

* Endpoint: /send
* Description: Test that a user cannot send a message to a user who has unmatched with them

CreateMatch

* Endpoint: /match
* Description: Test that matches between users can be created

CreateMatchNoProfile

* Endpoint: /match
* Description: Test that a user cannot match with a profile that has been deleted or does not exist

DeleteMatch

* Endpoint: /unmatch
* Description: Test that a user can unmatch with a matched user

GetMessages

* Endpoint: /retrieve
* Description: Test that a user can receive all messages between themselves and another user

**Auth Tests**

CreateAccount

* Endpoint: /register
* Description: Test that accounts can be successfully created

CreateInvalidAccount

* Endpoint: /register
* Description: Test that creating an account with invalid information fails

AccountLogin

* Endpoint: /login
* Description: Test that a user can successfully log in

AccountLoginBadCredentials

* Endpoint: /login
* Description: Test that a user cannot login to an account with incorrect login information

UpdateAccount

* Endpoint: /update
* Description: Test that a user can update their account

UpdateAccountWrongProfile

* Endpoint: /update
* Description: Test that a user cannot update another user’s account

DeleteAccount

* Endpoint: /delete
* Description: Test that a user can delete their account

DeleteAccountWrongProfile

* Endpoint: /delete
* Description: Test that a user cannot delete another user’s account