**Report on “Marriage Matchmaking App Assignment”**  
*By Debapriya Das*

**Approach:**   
In line with the provided tasks, I undertook the following steps:

1. Examined the entire codebase.
2. Created test users using the pre-existing routes provided.
3. Added the following routes:
   * **/users/update/{user\_id}**: Allows updating a user with the specified user\_id using the PUT method. Only the fields to be updated need to be provided in JSON format.
   * **/users/delete/{user\_id}**: Enables deletion of a user with the specified user\_id using the DELETE method.
   * **/users/matches/{user\_id}**: Finds potential matches for the user associated with the provided user\_id. The file **./matches.py** contains the function definition for finding matching users.
   * **/users/validate-email**: Validates any email address provided in the request body.
     + Added a new schema **EmailRequest** in the **./schemas.py.**
     + The email format is first checked using regular expressions.
     + The domain name is then verified using dns.resolver to ensure it is valid and has MX records.
     + The file **./check\_email.py** contains the function definition for email validation.
4. For each route all the possible error-handling the status-code responses are implemented.

**Assumptions:** To complete the project, the following assumptions were made:

* **Finding Matches:**
  + Matchings are based solely on similar interests.
  + The matching algorithm identifies similar users of the opposite gender only.
* **Email Validation:**
  + An endpoint was added to validate any email address requested.
  + Email validation is also applied during the creation and updating of user data.

**Testing:**

* Added route **/users/populate** to create bulk users for testing purposes. This route accepts an array of users in JSON format as the request body.