Joshua Beck

CST-235

October 16th, 2022

Prof. Smithers

Domain Description

The domain to be modeled through a web application is a dating application. This webpage will allow the viewing of various profiles and profile contact information. This domain differs somewhat from other domains as there is not necessarily a necessarily separate product and user. There are many correlations between the user and product, especially in the database design, as they exist as different versions of the same object. Although the webpage should allow a user to create an account and not “join” the dating service, “joining” (posting their profile for others to see) will use much of the user information with a few select additions. This has been taken into consideration when defining the database.

Products

Each “product” (in this case, person) will include the profile picture, hobbies, education (if applicable) including bachelors, masters, and doctorates, languages spoken, compatibility score, previous relationships, hair color, eye color, and height. The product record (labeled as DatingUser) will include a foreign key to a user object containing more personal information such as first name, last name, phone number, email, and address. The hobbies table will be separate and referenced by foreign key allowing a few-to-many relationship and will have a title and a description. The address field will have a one-to-one relationship to the user table and will contain elements such as address line 1, address line 2, city, state, country, and zip code. For more information on the product and product DB design, refer to the ER diagram in the project report.

Work Breakdown

When looking at the work to be performed, there is currently no team to divvy work out to so the person responsible for completing all the tasks is the student. However, the work has still been broken down using AGILE terminology. Forthcoming is outlined a few important elements upcoming in the issue queue, a few in progress, and a few completed issues.

To-dos

A product login and registration page need to be designed and implemented

A product list page needs to be designed and implemented

A project add page needs to be designed and implemented

A project edit/delete page needs to be designed and implemented

A database handler needs to be implemented

a database needs to be implemented

In-Progress

Designing a database and database relationships

Designing the models for the product

Defining the project scope and target audience

Creating wireframes

Complete

Defining the product

Creating an initial work breakdown

Initial documentation

Site map

Risk Plan

|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Details | Strategy for Risk Avoidance | Strategy for Mitigation |
| Limited audience adoption of product idea | Even in the case of a well-designed product, often user adoption of a particularly user-oriented services such as dating services runs the risk of being stunted. | To reconcile this, one must ensure that a clear target audience is identified and frequent soft test cases are performed against actual target customers. | Limit the services available to reorient towards a specific target audience. Once the target audience is reached, gradually release or reshape offered services |
| Device failure | Many hardware devices fail leading to lost data, corrupted code, and vanishing designs | Ensure that devices are up-to-date and that devices are recent and not outdated hardware. | Frequent backups to an external drive as well as frequent pushes of code to a GIT repository can mitigate this risk. |
| Overambitious project design | Many project ideas appear doable on paper but once implemented are significantly more time consuming and may not be possible due to time or resource limitations. | Submit project design to professor before implementation to get his professional opinion and break down each task with a time estimate | Divide time more intentionally with a teammate and leave weekends free as a cushion for an extra project load. |
| Corrupted Backend | Oftentimes, when pushing code to a repository, it causes elements on the frontend to be suddenly broken, inhibiting user experience | Do not submit code to the repository nor deploy code that is faulty | Ensure unit tests are available for all code, perform peer reviews of all code, do not deploy code to production without thoroughly testing develop |