Architecture Design

TEAM MYSTERY

TABLE OF CONTENTS

Non Relational Database Design

Database Design V1

Last updated

- August 20th

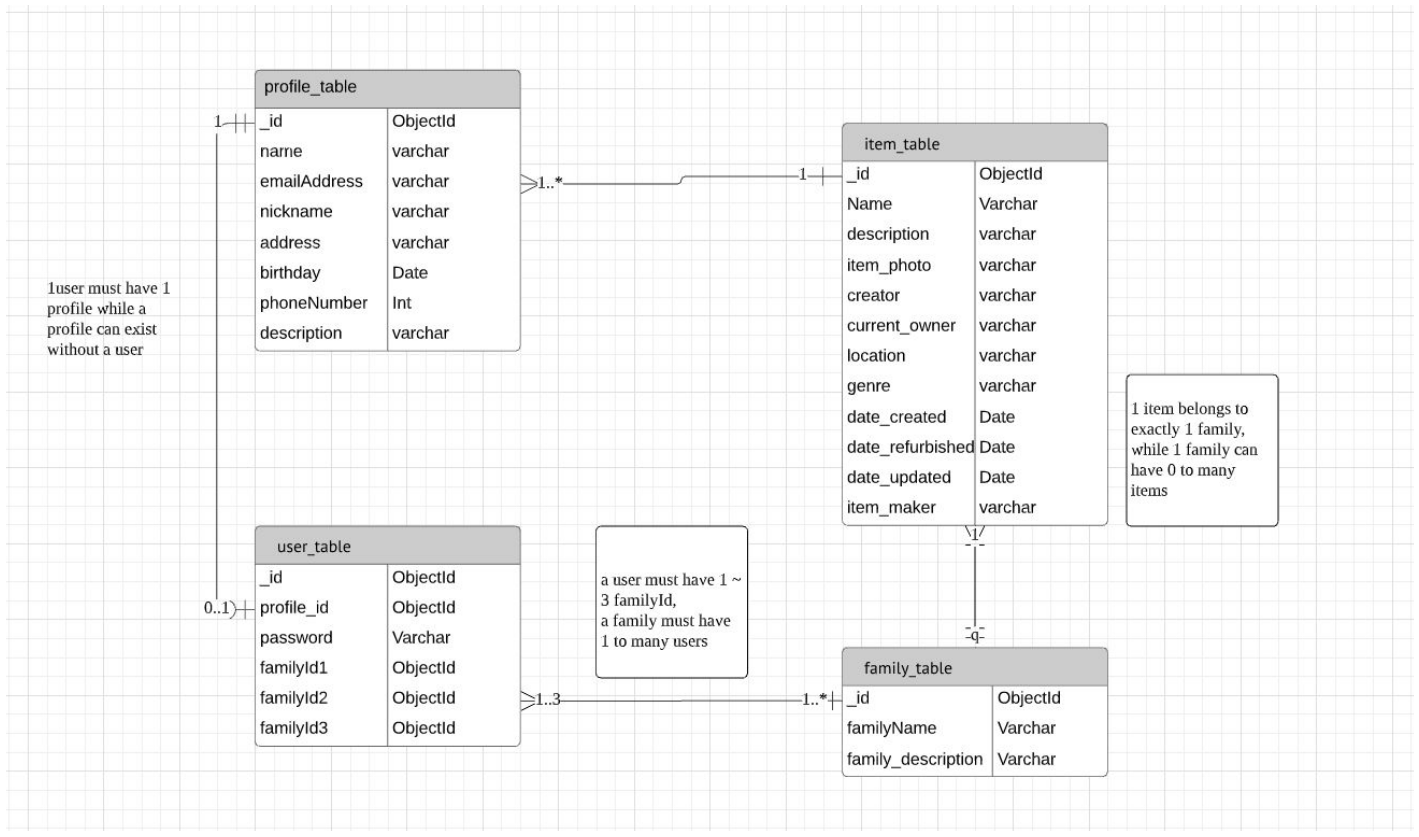
Sprint

- Sprint 1

Version Description

- This is our very first design made after collecting requirements from client
- Due to the nature of MongoDB, the tables actually don’t have physical relations, including multiplicities. However, here we drew lines between them to represent logical relations.

Database Design (names of tables could be discussed further)



profile_table	
_id	ObjectId
name	varchar
emailAddress	varchar
nickname	varchar
address	varchar
birthday	Date
phoneNumber	Int
description	varchar

item_table	
_id	ObjectId
Name	Varchar
description	varchar
item_photo	varchar
creator	varchar
current_owner	varchar
location	varchar
genre	varchar
date_created	Date
date_refurbished	Date
date_updated	Date
item_maker	varchar

user_table	
_id	ObjectId
profile_id	ObjectId
password	Varchar
familyId1	ObjectId
familyId2	ObjectId
familyId3	ObjectId

family_table	
_id	ObjectId
familyName	Varchar
family_description	Varchar

1user must have 1 profile while a profile can exist without a user

a user must have 1 ~ 3 familyId, a family must have 1 to many users

1 item belongs to exactly 1 family, while 1 family can have 0 to many items

Figure 1 Database design tables

Database Table Details

1. **Item_tables**: all treasure items, items of the same family are grouped by family_id
   id int notnull pk
   Name varchar
   Description varchar
   Item_photo url varchar
   date _added date default notnull The date that item is added to the database
   Date_updated date default
   Date_created data default notnull The date that the item was created(a picture shot by client/ an item purchased by client)
   Date_refurbished data default The data the the item was refurbished
   Creator varchar notnull
   Current_owner varchar notnull
   Location varchar notnull (str 'online' for digital treasure)
   Genre varchar notnull
   Family_id int notnull

2. **profile_tables**: all users, users of the same family are grouped by family_id(here we assume that a person who has place in more than one family would have several entries in the database, with different family_id and user_id)
   profile_table and item_table can join by i.creator = u.name or i.current_owner=u.name
   profile_id int notnull pk
   email_address pk,which can be used to login
   name varchar notnull
   nickname varchar notnull
   Family_id int notnull
   Address varchar
   Phone_number int
   Birthday Date
   Authorization bool

3. **family_tables**:
   Family_id int notnull
   Family_name varchar notnull

4. **account_tables**:
   All users have profile while not all profiles have corresponding user(e.g passed people)
   profile_table and user_table can join by p.user_id = u.user_id
   User_id int
   Password varchar
   Family_id1 int

Family_id2 int
Family_id3 int

Non Relational Database Design V2

Last updated

- September 10th

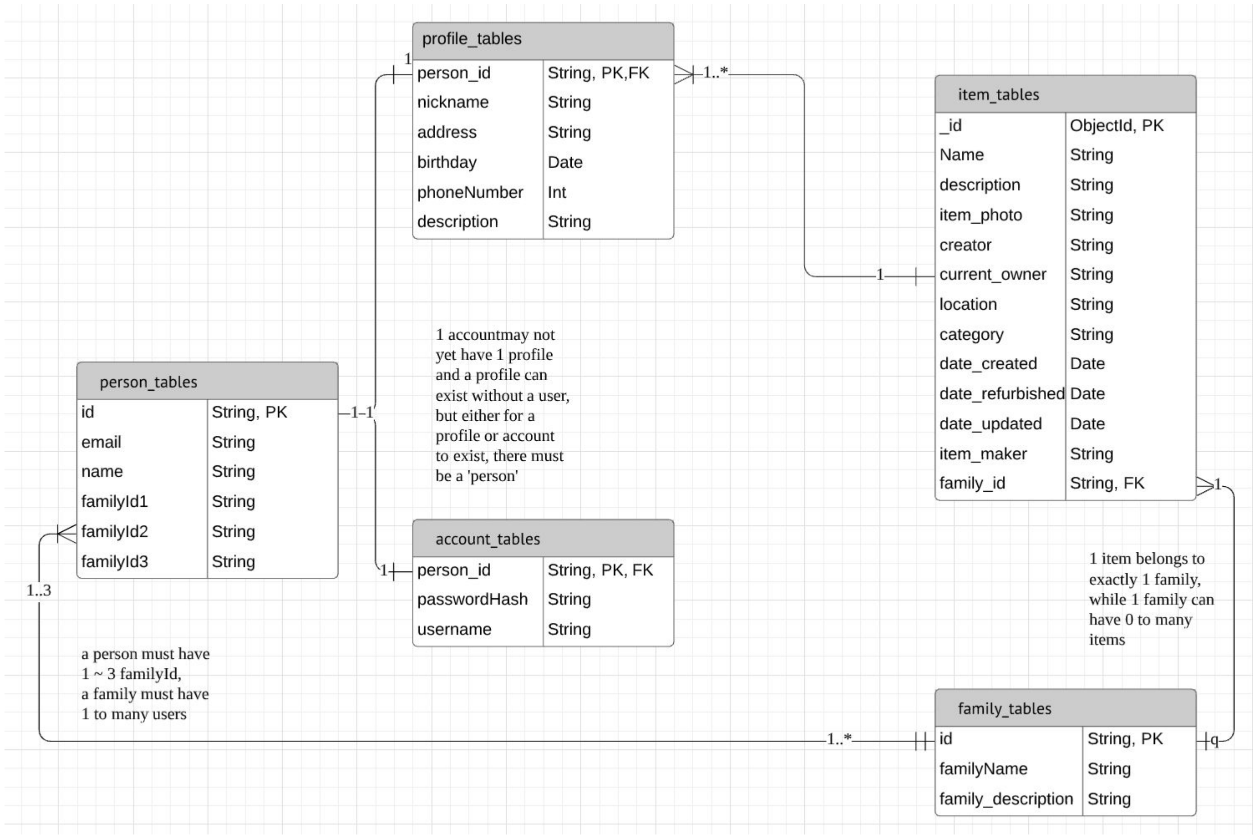
Last version

- File ‘Non Relational Database Design v1’, under the same folder https://docs.google.com/document/d/1cb-047RQjQOZZ-SqPYB8viTdjZNY4KYNVQmI2OxwZgI/edit?usp=sharing

New Features

1. All table names now end with ‘s’ to be suitable for mongoDB
2. Changes of attributes including fields, names, and data types have been updated
3. Addition of person_tables, to generalise fields that profile_tables and account_tables share in common

Database Design (names of tables could be discussed further)



profile_tables	
person_id	String, PK,FK
nickname	String
address	String
birthday	Date
phoneNumber	Int
description	String

item_tables	
_id	ObjectId, PK
Name	String
description	String
item_photo	String
creator	String
current_owner	String
location	String
category	String
date_created	Date
date_refurbished	Date
date_updated	Date
item_maker	String
family_id	String, FK

person_tables	
id	String, PK
email	String
name	String
familyId1	String
familyId2	String
familyId3	String

account_tables	
person_id	String, PK, FK
passwordHash	String
username	String

family_tables	
id	String, PK
familyName	String
family_description	String

1 accountmay not yet have 1 profile and a profile can exist without a user, but either for a profile or account to exist, there must be a 'person'

1 item belongs to exactly 1 family, while 1 family can have 0 to many items

a person must have 1 ~ 3 familyId, a family must have 1 to many users

Figure 2 Database design tables

Non Relational Database Design V3

Link

https://www.lucidchart.com/invitations/accept/90fcd97b-a4cd-4443-8494-ad66918fba5b

Last updated

- October7th

Last version

- File ‘Non Relational Database Design v2’

New Features

1. Add new tables to store photos
   - Why not just add ‘path’ as attributes in corresponding tables as they are 1-to-1 relation?
     <u>To make it easy to create photo object in code , as we use fs.writeFileSync</u>
   - Why not put familyPhoto and profilePhoto at the same table, since all of their attributes have the same data types?
     <u>In case of some families’ ids are the same as some users’ ids and bring confusion and inconvenience</u>
2. Birthdays are now stored as year/month/day, for 3 benefits: 1. Reduce confusion to users when they need to enter a date, 2. Clearer in database, 3. Easier to sort and to be shown at frontend. The date attribute of items stay String, to give users flexibility with items with obscure dates
3. Remove the person table which contains the shared information of profile_tables and user_tables, as profile_tables and user_tables actually barely overlap
4. Delete relation between profile and user, as a part of reducing scope

Database Design

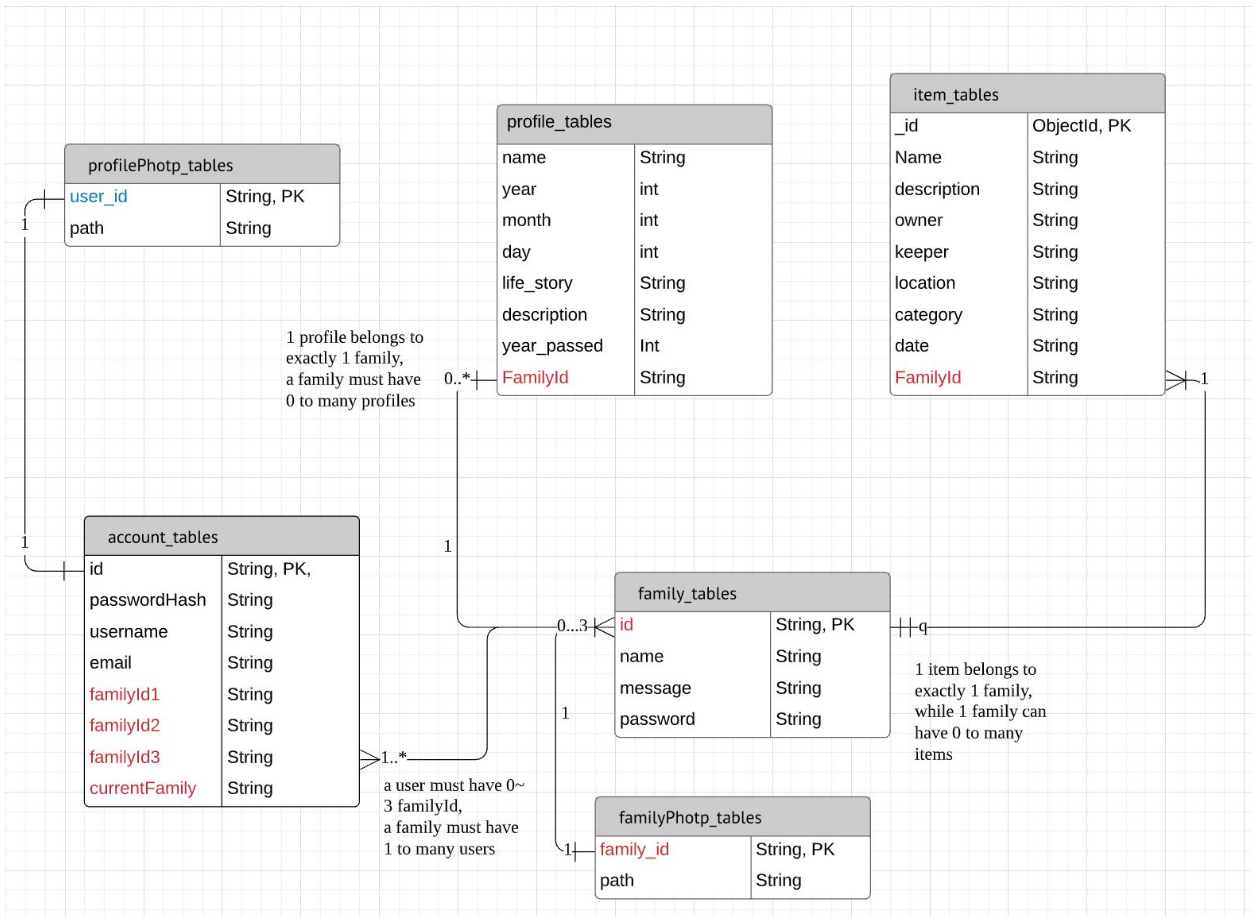


Figure 3 Database design tables

System Flow Design(substitutes SSD)

Flow Chart Link

- Flow chart of the website's supposed jumping between pages(substitution of SSD): https://www.lucidchart.com/invitations/accept/264c3994-ede6-4af0-bd95-cef3ac3fecba

Last Updated

- September 27th

Notations

1. an arrow means jumpable from one page to the other
2. pages on navigation bar can be reached by clicking on the navigation bar
3. dash line refers to corresponding functions
4. colors: orange = error, blue = user input page, green = information showing page
5. green line = when operation successful, red line = when operation failed
6. arrow from left and top= in, arrow from right and bottom = out

Overall look

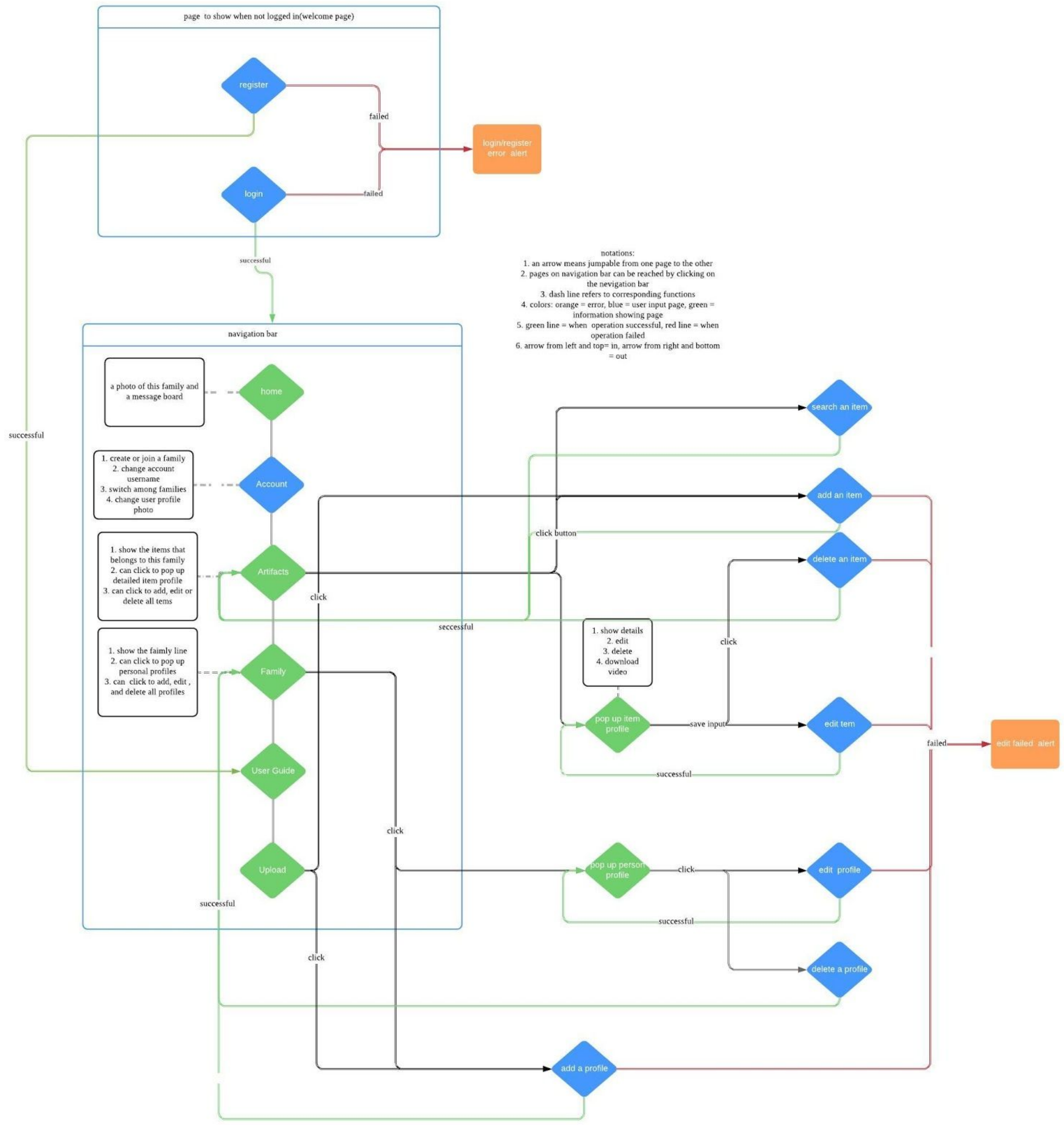

page to show when not logged in(welcome page)
register
login
failed
failed
login/register error alert
successful
successful
notations:
1. an arrow means jumpable from one page to the other
2. pages on navigation bar can be reached by clicking on the nevigation bar
3. dash line refers to corresponding functions
4. colors: orange = error, blue = user input page, green = information showing page
5. green line = when operation successful, red line = when operation failed
6. arrow from left and top= in, arrow from right and bottom = out
navigation bar
a photo of this family and a message board
home
1. create or join a family
2. change account username
3. switch among families
4. change user profile photo
Account
1. show the items that belongs to this family
2. can click to pop up detailed item profile
3. can click to add, edit or delete all tems
Artifacts
1. show the faimly line
2. can click to pop up personal profiles
3. can click to add, edit , and delete all profiles
Family
User Guide
Upload
search an item
add an item
click button
delete an item
click
seccessful
1. show details
2. edit
3. delete
4. download video
click
pop up item profile
save input
edit tem
successful
failed
edit failed alert
click
pop up person profile
click
edit profile
successful
delete a profile
successful
click
add a profile

Figure 1 The whole diagram

Welcome page zoom-in

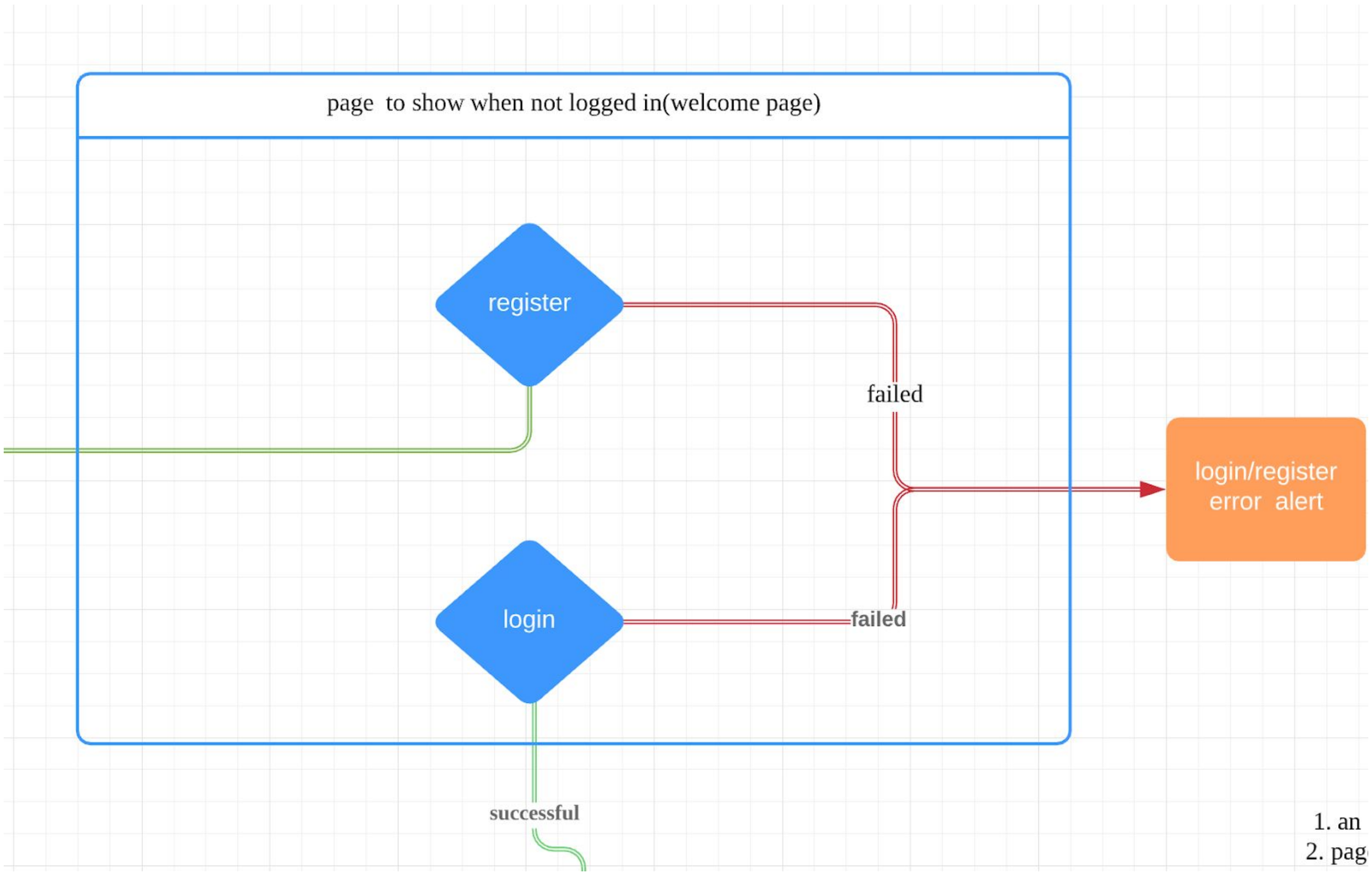

page to show when not logged in(welcome page)
register
failed
login/register error alert
login
failed
successful
1. an
2. pag

Figure 2 welcome page zoom-in

Navigation bar zoom-in

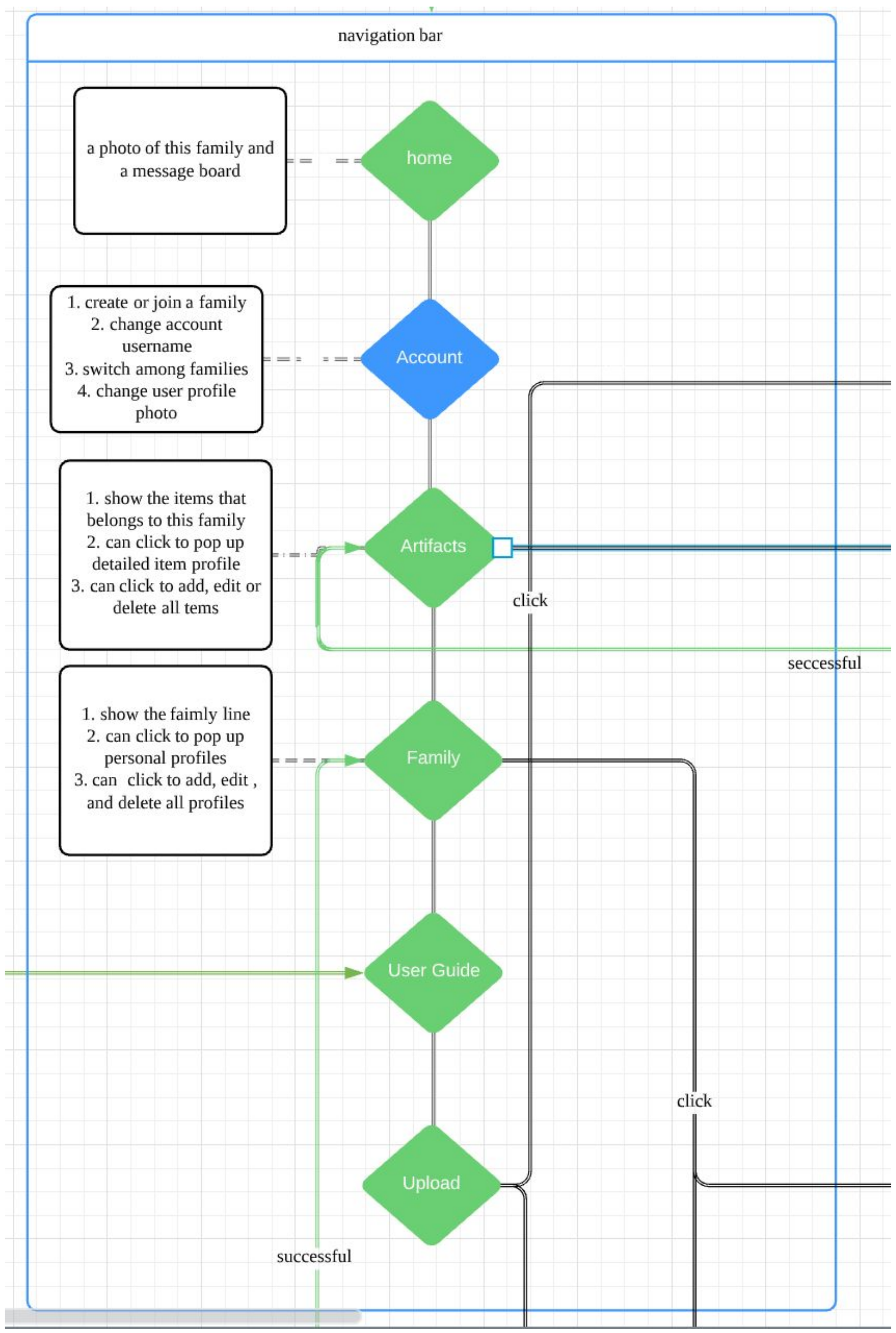

navigation bar
a photo of this family and a message board
home
1. create or join a family
2. change account username
3. switch among families
4. change user profile photo
Account
1. show the items that belongs to this family
2. can click to pop up detailed item profile
3. can click to add, edit or delete all tems
Artifacts
click
seccessful
1. show the faimly line
2. can click to pop up personal profiles
3. can click to add, edit , and delete all profiles
Family
User Guide
click
Upload
successful

Figure 3 navigation bar zoom-in

Artifacts related functions flow zoom-in

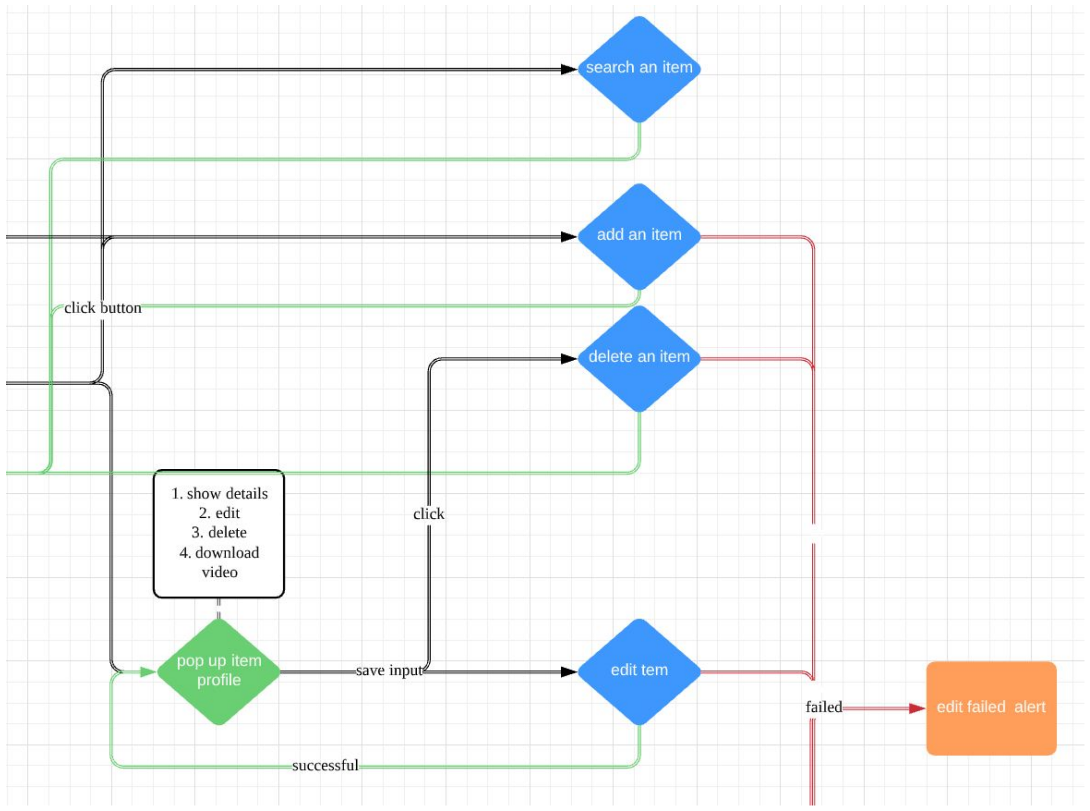


Figure 4 Artifacts related functions flow zoom-in

Profiles related functions flow zoom-in

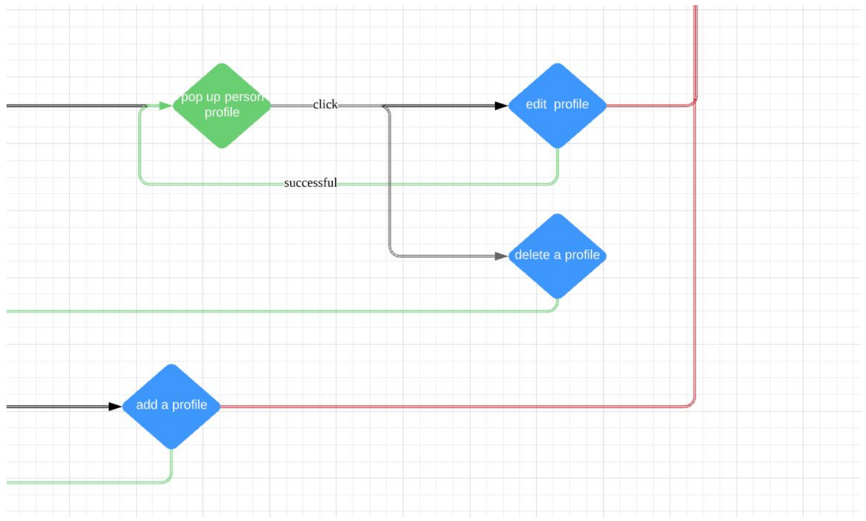
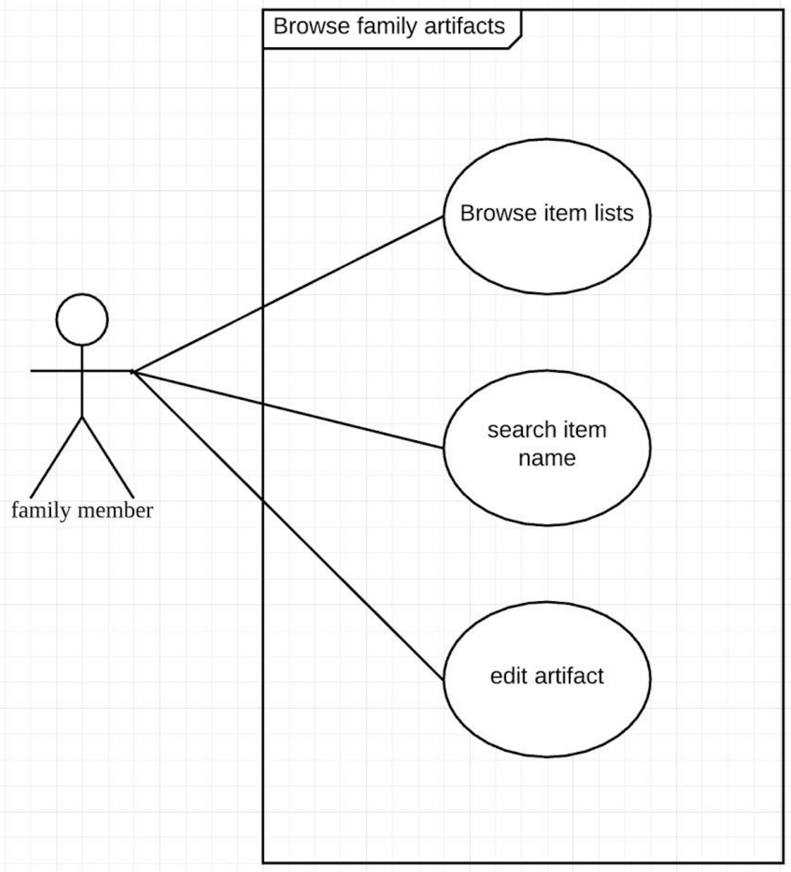


Figure 5 Profiles related functions flow zoom-in

Use Case Diagram

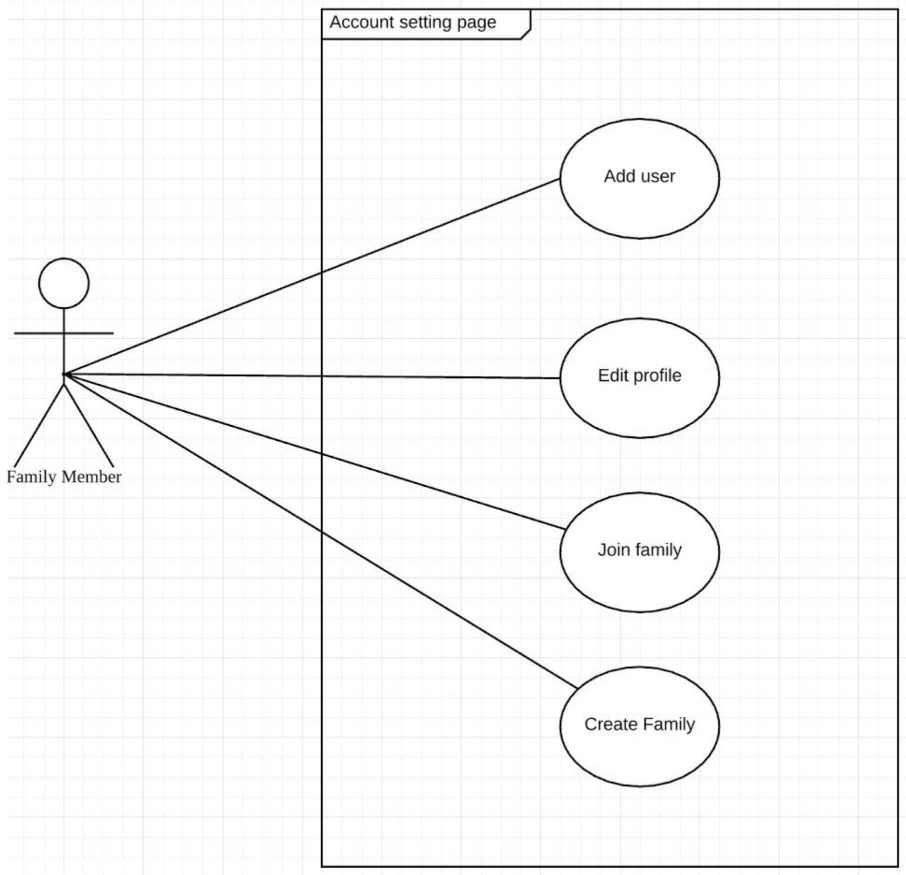
Browse family artifacts



Main Success Scenario

Step	Actor	Action Description
1	family member	artifacts are displayed in this page
2	family member	enter words in search bar, and result items show
3	family member	family member can click “edit” bottom
4	family member	family member can successfully edit the artifacts
5	family member	shows submit success page

Account setting page

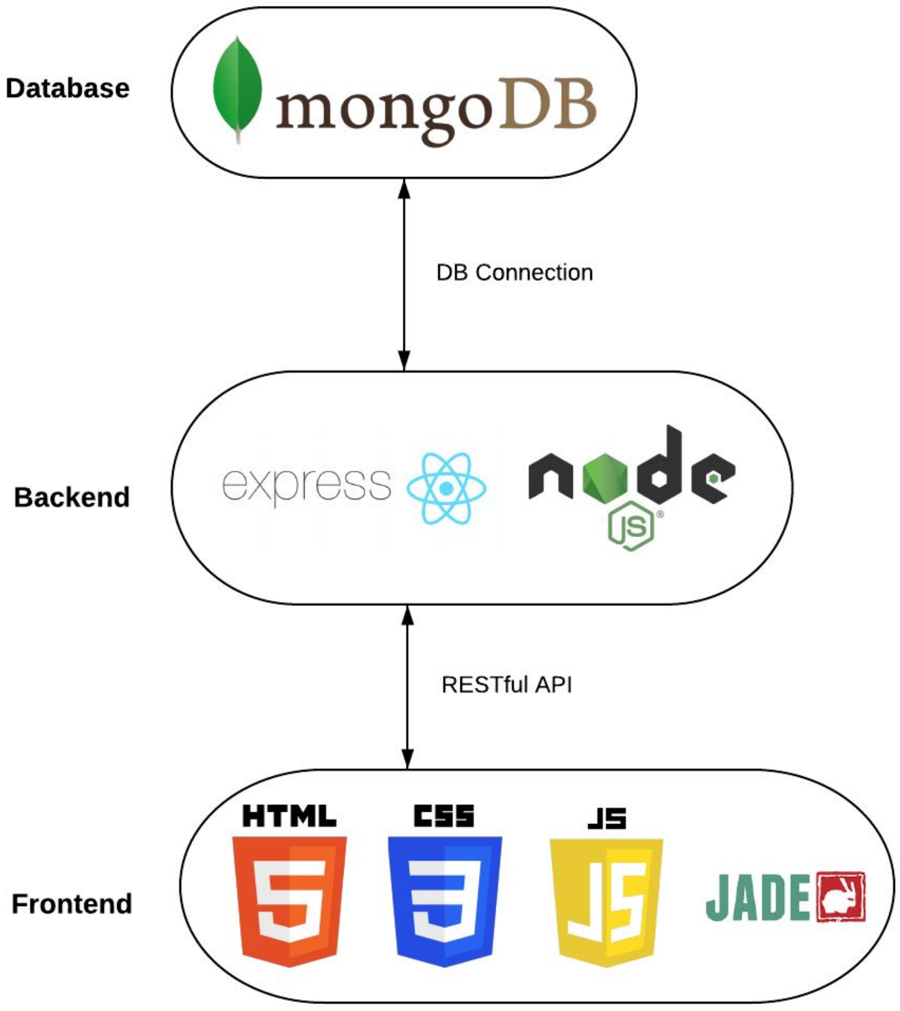


Main Success Scenario

Step	Actor	Action Description
1	family member	click “add user” bottom and show ”add user” page
2	family member	family member can submit the user profile
3	family member	family member can edit their profile
4	family member	can join family with family ID
5	family member	can create family successfully

Tech Stack

Tech Stack Diagram



Tech Stack Diagram

Structure

- HTML
    - HTML is widely used in writing website, and almost every browser supports HTML language.
    - It is easy to learn and to use.
    - It can display changes instantly by reloading the previous HTML page.
    - It can create only static and plain pages so if we need dynamic pages then HTML is not useful.

- Jade
    - Jade can create dynamic pages rather than HTML.
    - Tags used in jade is simple, which could make the code look concise and structured.
    - It is not convenient to display changes, we have to rerun the whole project after doing some changes.

Presentation

- CSS
    - It is easy to read, to learn and to use.
    - It is compatible with HTML and Jade.
    - CSS can save time. By making one change to css style sheet, we can automatically make it to every page we would like to update.

Behaviour

- Javascript
    - It is easy to learn and offers syntax close to English.
    - It uses DOM model which provides many predefined functionalities to the various objects on pages making it a breeze to develop a script to solve a custom propose.
    - No compiler is needed, the browser can interpret it.

Runtime Environment

- Node.js
    - It is popular, and supported very well.
    - It is implemented on top of the Google Chrome’s V8 engine which is super fast.
    - It does I/O better.
    - Files I/O and db queries are non-blocking.

    - It has great performance even handling a big number of data.

Frameworks

- Express.js
    - It streamlined node.
    - It allows us to define routes based on HTTP methods and URLs.
    - It includes many middleware modules to perform tasks on request and response.
    - It allows us to create RESTful API server.
    - It is easy to connect with database.

Database

- MongoDB
    - It uses BSON format which enables to internally index and map document properties.
    - It does not require data structures, that are unified in nature across all the objects that are being used, which makes it convenient to use.
    - It can fast and easy process data.