# B. Tech Project Presentation

MID TERM ASSESSMENT

BY APOORV TOMAR

141100050

## Objective And motivation

The main objective of the project is to develop a face-based search web application. The application will allow users to create profile and upload face images to explore another user's profile. It will use deep learning face recognition algorithm to perform face matching.

The project could be used as backend for optical head mounted display devices to detect face of others user on Realtime basis.

Due to limited availability and high cost of such devices, The project will be prototyped on web application

## Literature Review

## Face Detection Algorithm

## Viola Jones Algorithm

- A seminal approach to real-time object detection.
- Training is slow, but detection is very fast
- Key ideas:

-> Haar Feature Selection

-> Creating an Integral Image

-> Adaboost Training

-> Cascading Classifiers

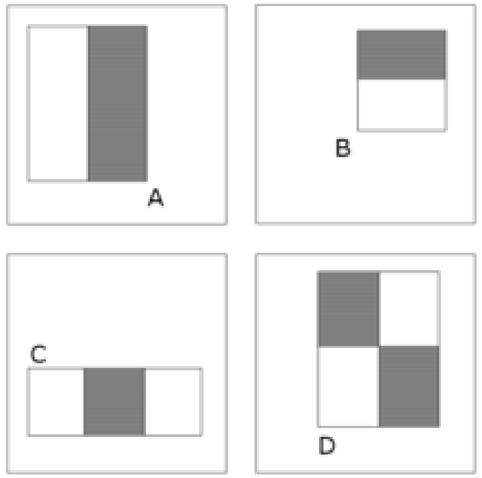
## Haar Features

Haar basis functions consist of:

• two-rectangle feature: difference between the sum of the pixels within two rectangular regions

• three-rectangle feature: sum within two outside rectangles subtracted from the sum in a center rectangle

• four-rectangle feature: difference between diagonal pairs of rectangles.


A
B
C
D

## Contd.

Viola jones algorithm uses 24 x 24 pixels sub-window is applied of such features all over the image.

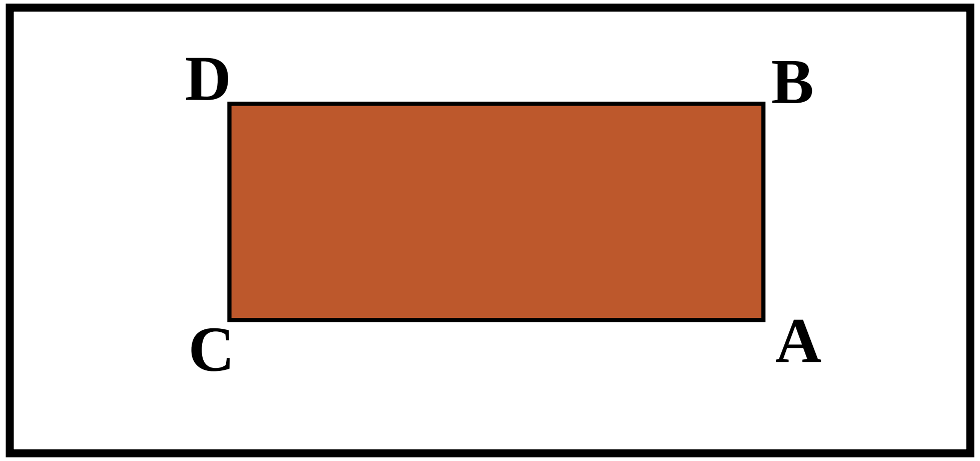
Consider all types of features present in the this 24 x 24 pixel sub-window. We can obtain more than 160,000 features in this sub-window.

Hence we calculate integral image.

## Computing sum within a rectangle

- Let A,B,C,D be the values of the integral image at the corners of a rectangle
- Then the sum of original image values within the rectangle can be computed as:

  sum = A – B – C + D
- Only 3 additions are required for any size of rectangle!



## Feature selection

- For a 24x24 detection region, the number of possible rectangle features is ~160,000!
- At test time, it is impractical to evaluate the entire feature set
- Can we create a good classifier using just a small subset of all possible features?
- How to select such a subset?

# Boosting

- *Boosting* is a classification scheme that combines *weak learners* into a more accurate *ensemble classifier*
- Weak learners <u>based</u> on rectangle filters:

value of rectangle feature

$$h_t(x) = \begin{cases} 1 & \text{if } p_t f_t(x) > p_t \theta_t \\ 0 & \text{otherwise} \end{cases}$$

window

parity

threshold

- Ensemble classification function:

$$C(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \alpha_t h_t(x) > \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases} \quad \longleftarrow$$

learned weights

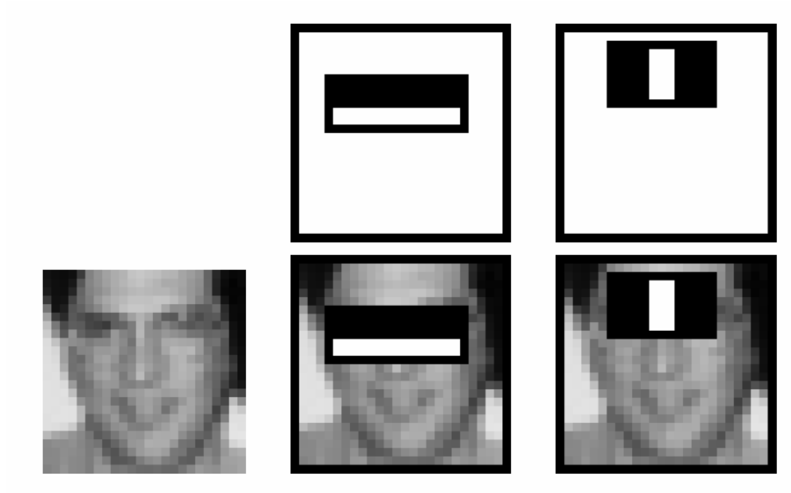
## Training procedure

- Initially, weight each training example equally
- In each boosting round:
  - Find the weak learner that achieves the lowest *weighted* training error
  - Raise the weights of training examples misclassified by current weak learner
- Compute final classifier as linear combination of all weak learners (weight of each learner is directly proportional to its accuracy)
  - Exact formulas for re-weighting and combining weak learners depend on the particular boosting scheme (e.g., AdaBoost)

Y. Freund and R. Schapire, A short introduction to boosting, *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, September, 1999.

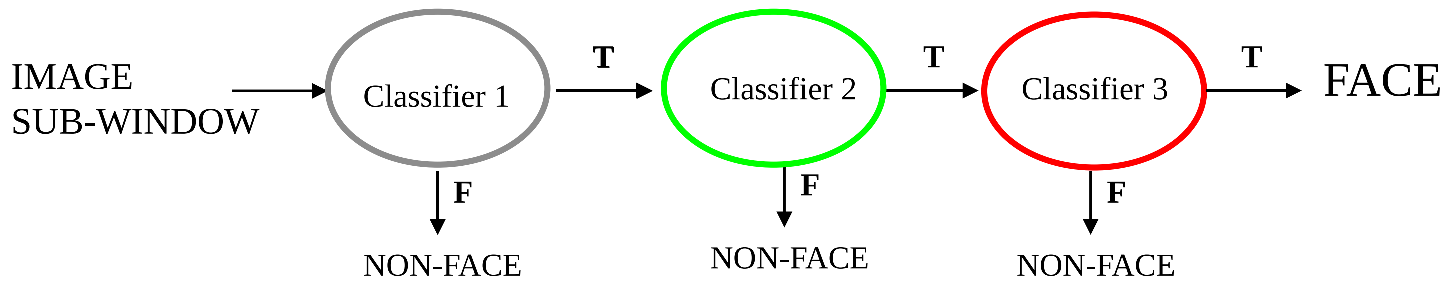
## Boosting for face detection

- First two features selected by boosting:



# Attentional cascade

- We start with simple classifiers which reject many of the negative sub-windows while detecting almost all positive sub-windows
- Positive response from the first classifier triggers the evaluation of a second (more complex) classifier, and so on
- A negative outcome at any point leads to the immediate rejection of the sub-window



# Training the cascade

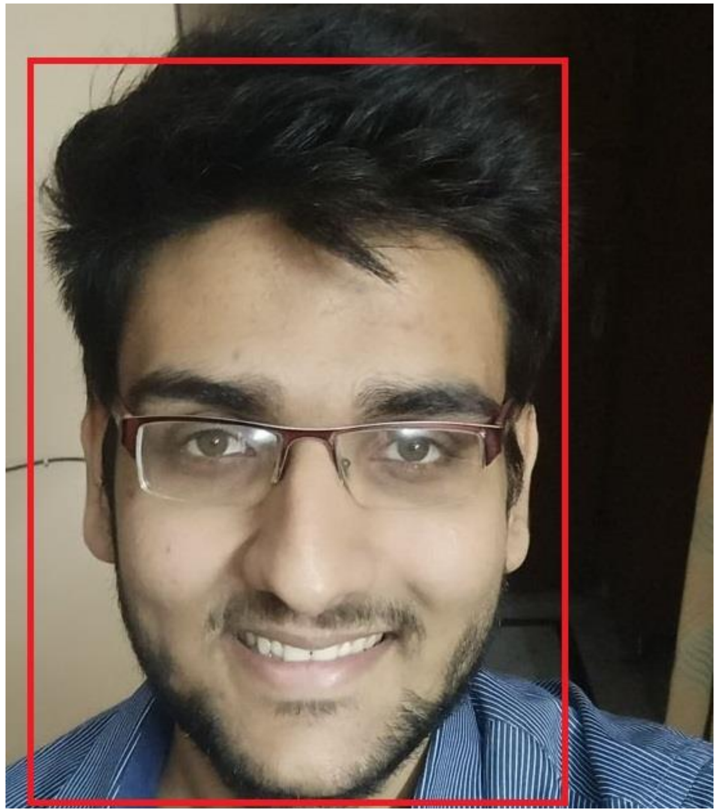
- Set target detection and false positive rates for each stage
- Keep adding features to the current stage until its target rates have been met
  - Need to lower AdaBoost threshold to maximize detection (as opposed to minimizing total classification error)
  - Test on a *validation set*
- If the overall false positive rate is not low enough, then add another stage
- Use false positives from current stage as the negative training examples for the next stage

# The implemented system

- Training Data
  - 5000 faces
    - All frontal, rescaled to 24x24 pixels
  - 300 million non-faces
    - 9500 non-face images
  - Faces are normalized
    - Scale, translation
- Many variations
  - Across individuals
  - Illumination
  - Pose

- Viola Jones is still most commonly used algorithm for purpose of face detection.
- In the original research paper, it showed 97% accuracy in detecting faces from image dataset.
- Researchers are now exploring face detection algorithms using neural networks.
- Histogram of Oriented Gradients HOGs can be used to detect face pattern via deep learning.

## Sample Photo

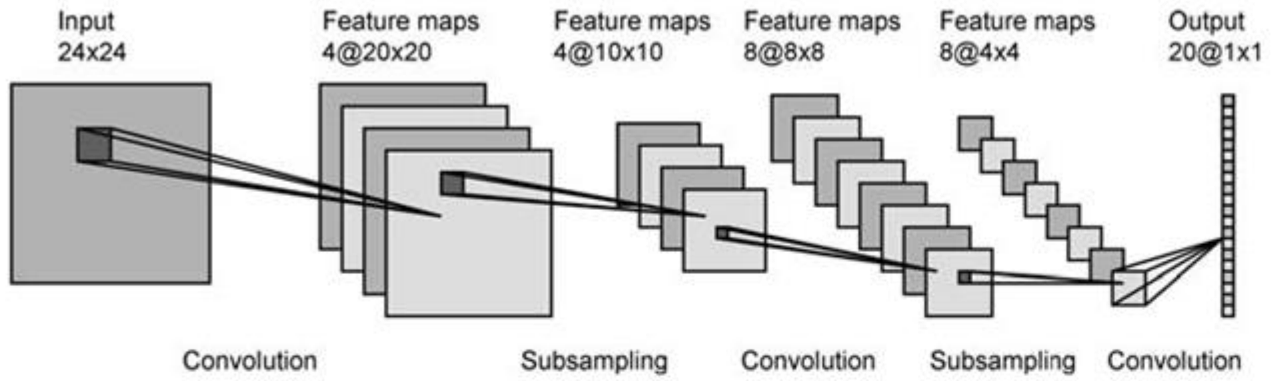


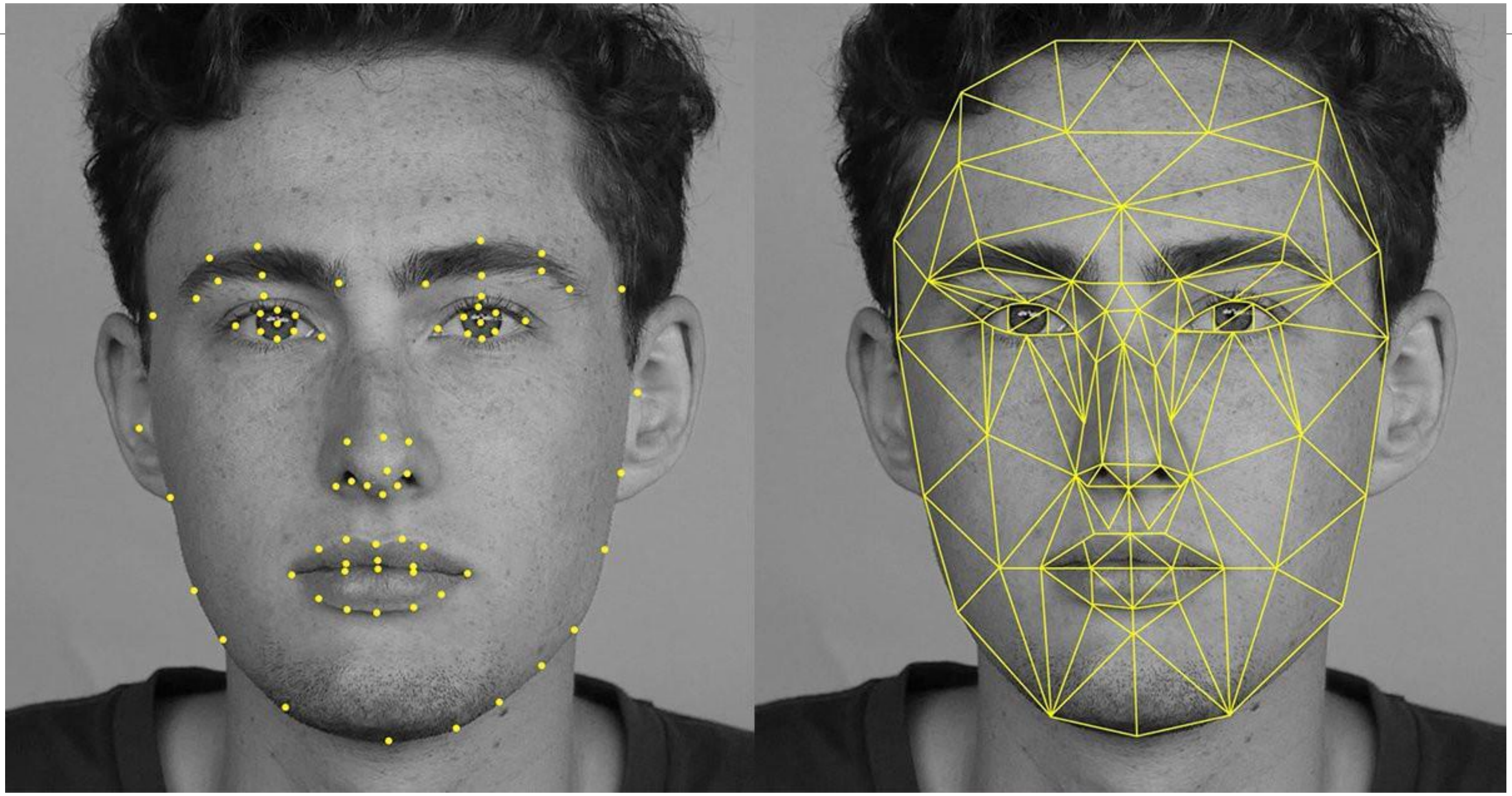
## Face Recognition

## Face Recognition using Deep Learning

- Deep Convolutional Neural Networkss are used to correctly recognize the face image from image database.
- DEEPFace and OPENFace are openly available neural networks system which have been trained over millions of face images.
- A face image is passed through this neural networks to obtained 128 measurements which uniquely define a face image.
- The 128 measurements are compared with existing image dataset measurements.
- The image measurements which measure closest to the obtained measurements. That image is matched with the input image.

# Sample convolution network




Face Landmarks detection

## Cont.

Since, storing images and training classifiers using deep neural networks is computationally expensive.

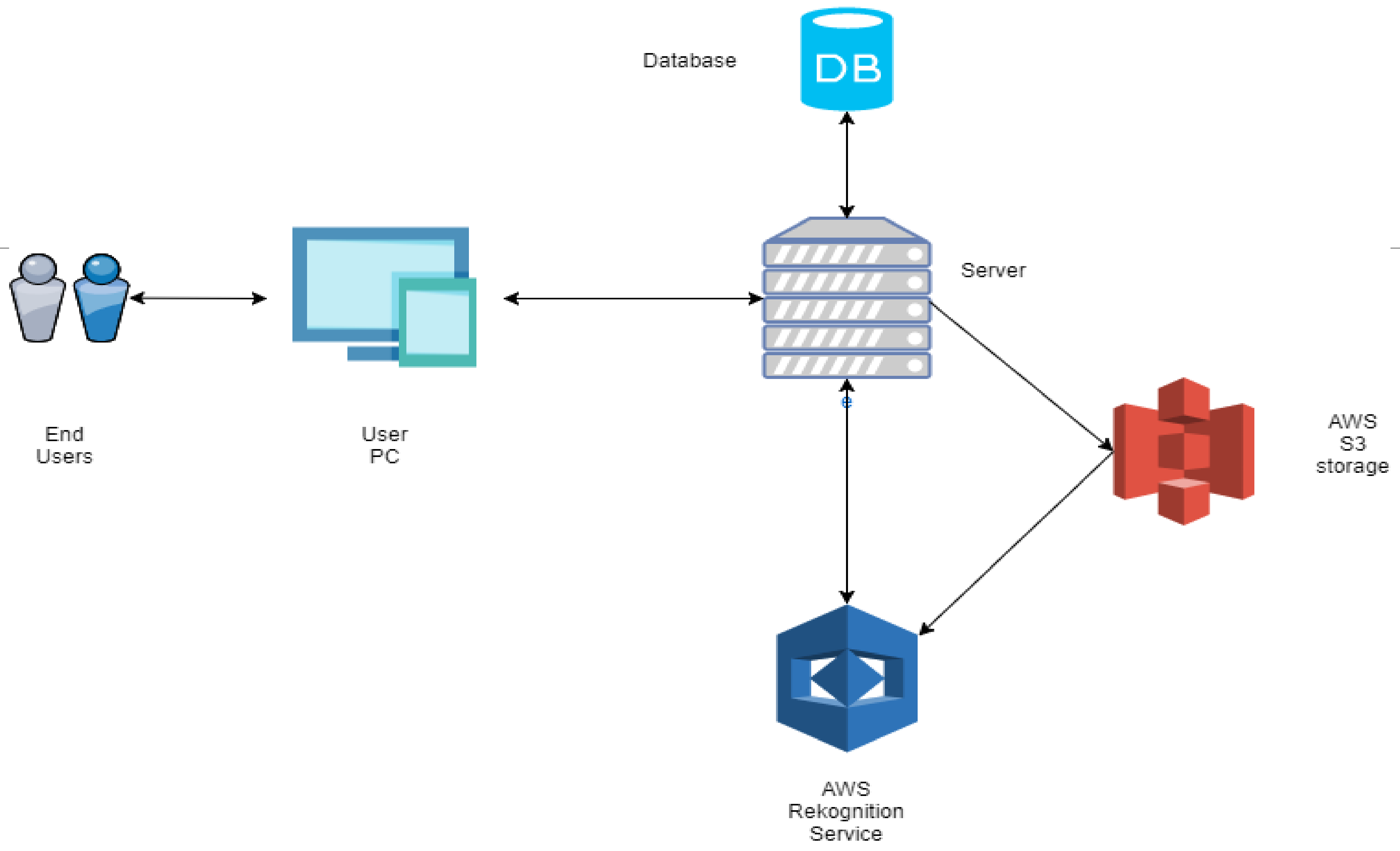
I have leveraged Amazon Web Service’s ‘S3 storage’ and ‘Rekognition’ to store images and perform facial recognition.

The Rekognition service is highly scalable, facial analysis and facial recognition service.

## Implementation

## Technologies used

- Ruby
- Sinatra library
- MySQL
- HTML, CSS, JavaScript
- Amazon Web Services: S3 Storage, Rekognition Service.



System Diagram

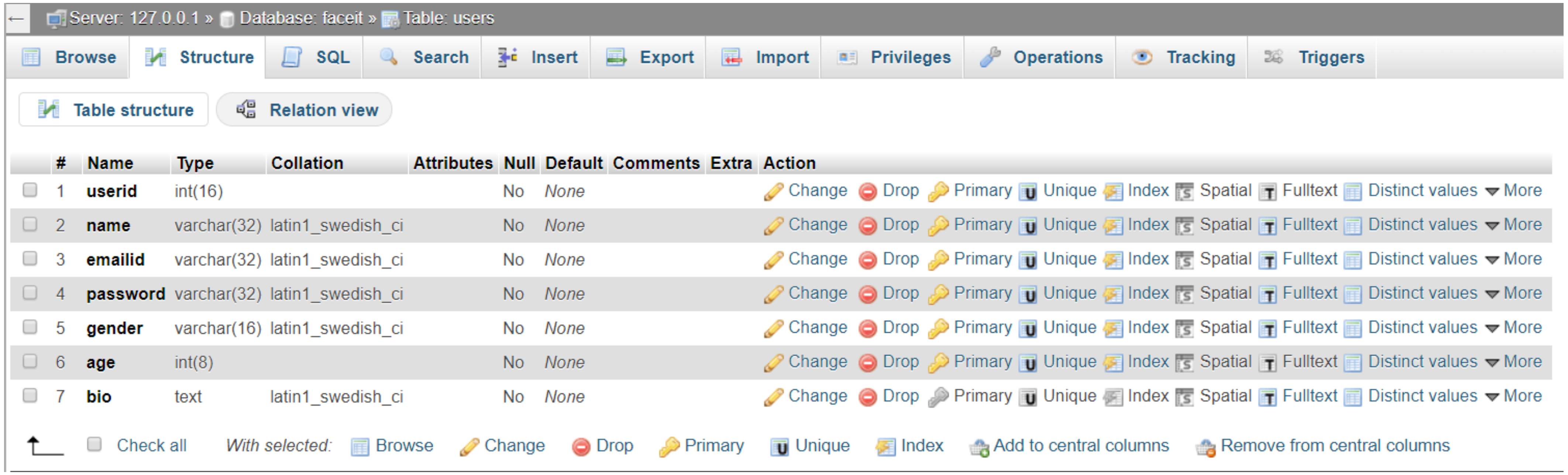
# Methodology

- A web application using Sinatra library on ruby is deployed on localhost machine.
- The web application asks for user permission to use camera sensor present in device.
- New collection is created in Amazon S3 storage service to store images.
- The user uploads his face image and assign his/her name to the photo.
- The photo is sent using HTTP PUT request to the S3 storage endpoint.
- The image is then used as face image input to Rekognition service. Neural network is trained over new input images.
- The user presses compare face image on the web application. If the image contains face images. The web app forwards the image to Rekognition service through API call.

## Contd.

- If the image does not contain any face, “Image does not contain face” is displayed on the HTML page.
- If the image contains face, Most matched face’s label and confidence is returned from the cloud computer to the main server in JSON format.
- The matched image label and confidence value is displayed over the HTML page.

# FaceIt Database Schema



Server: 127.0.0.1 » Database: faceit » Table: users

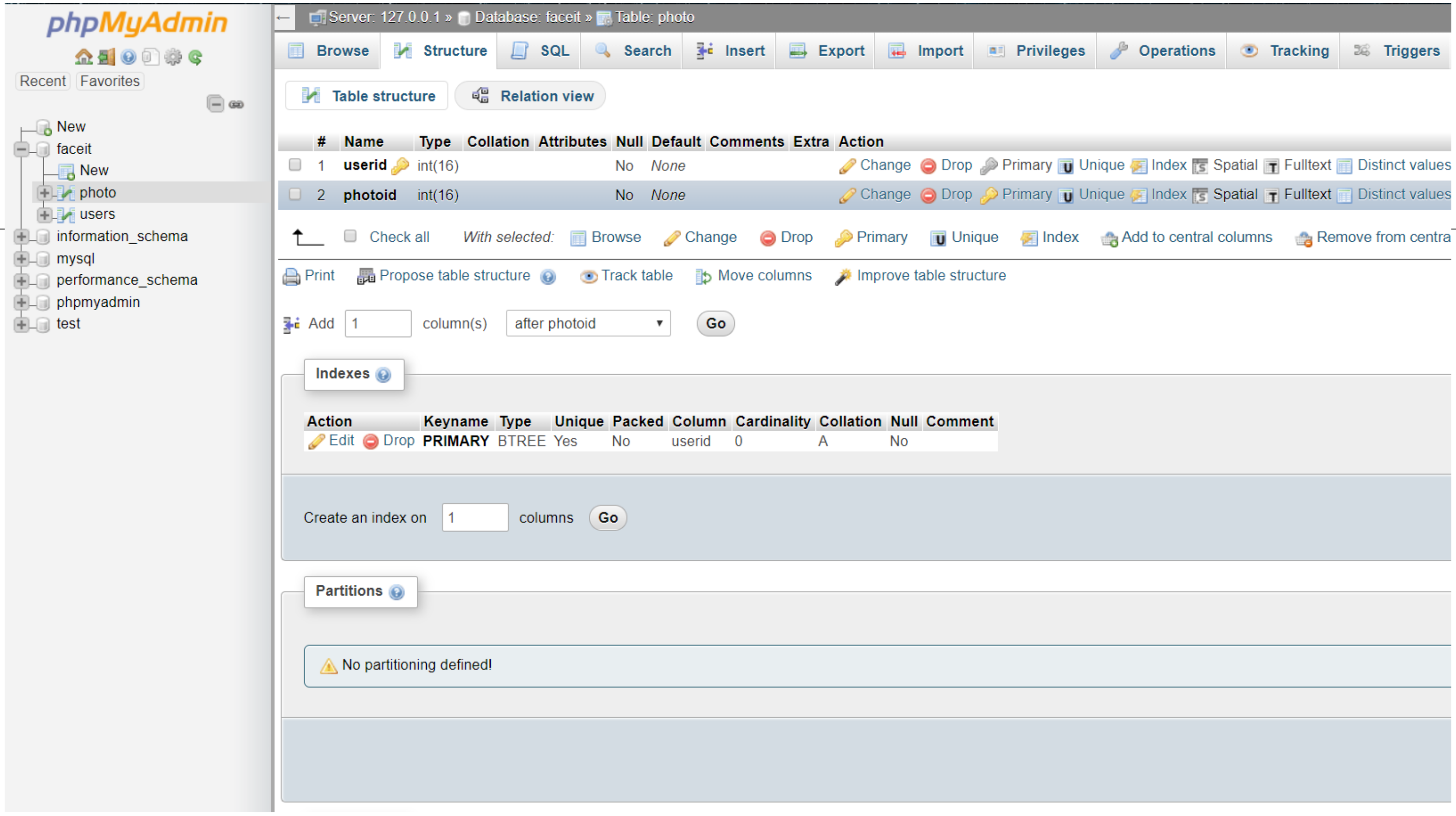
Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Tracking | Triggers

Table structure | Relation view

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|
| 1 | **userid** | int(16) | | | No | *None* | | | Change Drop Primary Unique Index Spatial Fulltext Distinct values More |
| 2 | **name** | varchar(32) | latin1_swedish_ci | | No | *None* | | | Change Drop Primary Unique Index Spatial Fulltext Distinct values More |
| 3 | **emailid** | varchar(32) | latin1_swedish_ci | | No | *None* | | | Change Drop Primary Unique Index Spatial Fulltext Distinct values More |
| 4 | **password** | varchar(32) | latin1_swedish_ci | | No | *None* | | | Change Drop Primary Unique Index Spatial Fulltext Distinct values More |
| 5 | **gender** | varchar(16) | latin1_swedish_ci | | No | *None* | | | Change Drop Primary Unique Index Spatial Fulltext Distinct values More |
| 6 | **age** | int(8) | | | No | *None* | | | Change Drop Primary Unique Index Spatial Fulltext Distinct values More |
| 7 | **bio** | text | latin1_swedish_ci | | No | *None* | | | Change Drop Primary Unique Index Spatial Fulltext Distinct values More |

Check all *With selected:* Browse Change Drop Primary Unique Index Add to central columns Remove from central columns

User Table



Server: 127.0.0.1 » Database: faceit » Table: photo

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra |
|---|---|---|---|---|---|---|---|---|
| 1 | userid | int(16) | | | No | *None* | | |
| 2 | photoid | int(16) | | | No | *None* | | |

Add 1 column(s) after photoid

Indexes

| Keyname | Type | Unique | Packed | Column | Cardinality | Collation | Null | Comment |
|---|---|---|---|---|---|---|---|---|
| PRIMARY | BTREE | Yes | No | userid | 0 | A | No | |

Create an index on 1 columns

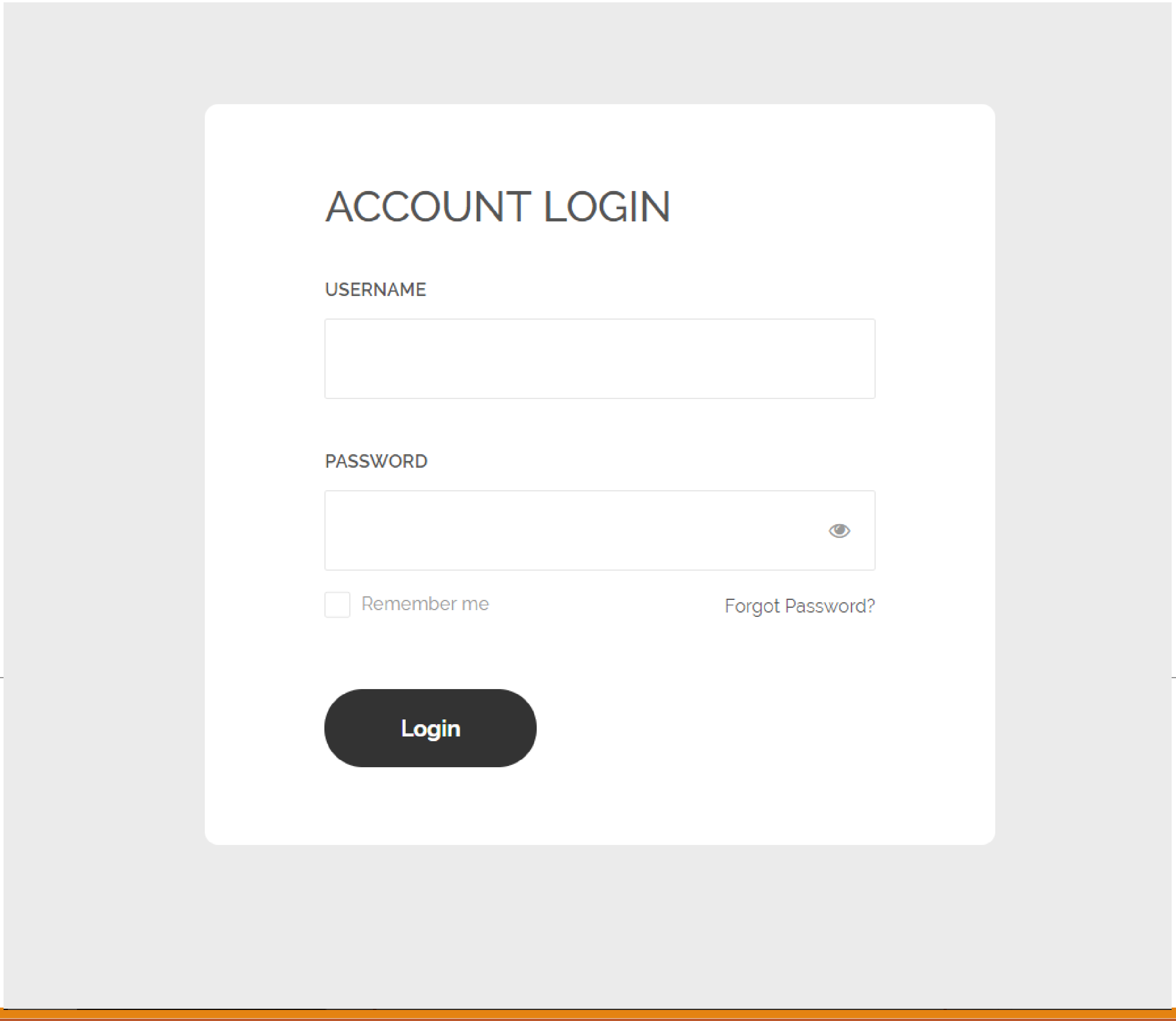
Partitions

No partitioning defined!

Photomatch table

## Web Platform UI

## Login Page


ACCOUNT LOGIN

USERNAME

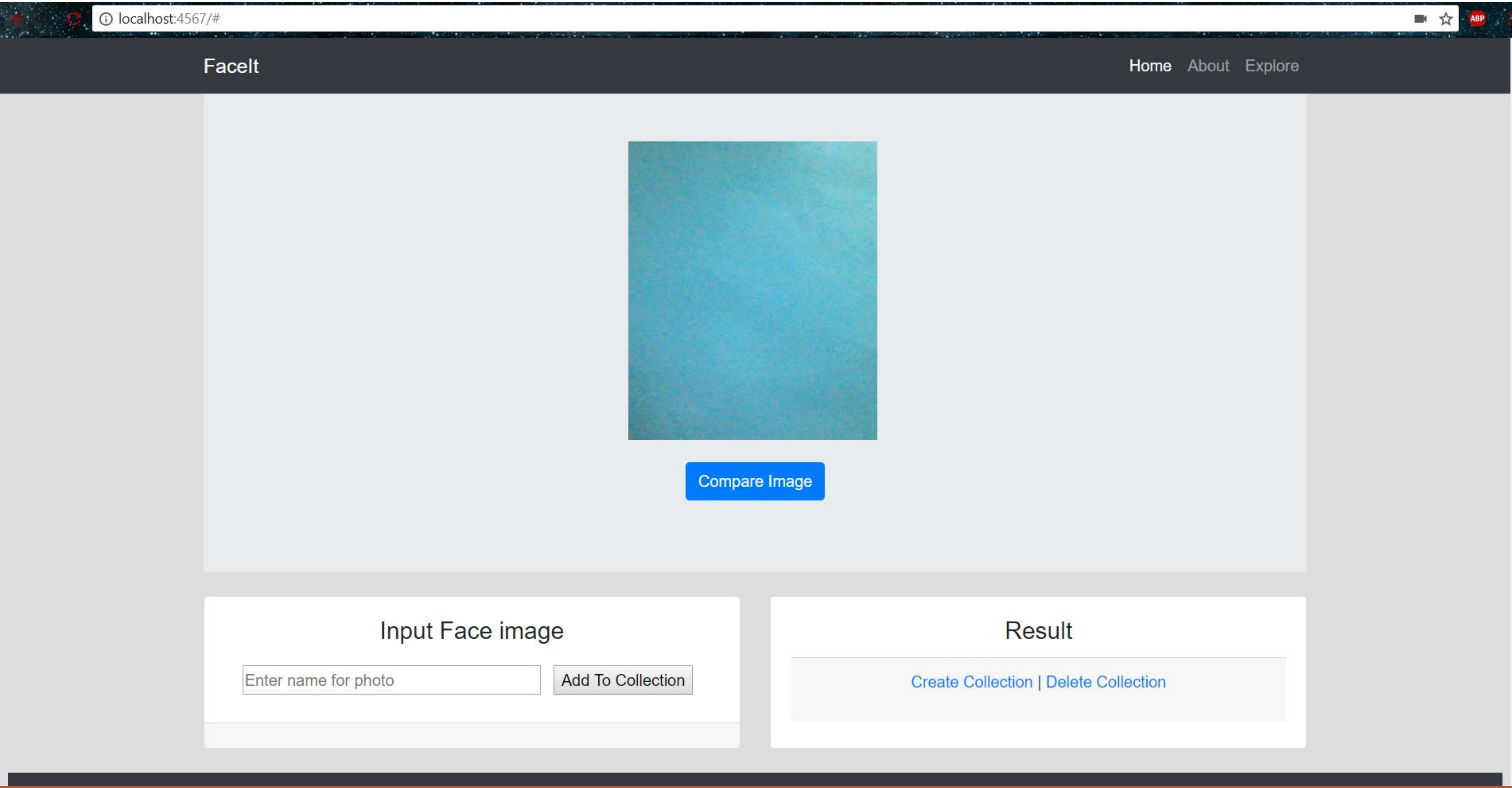
PASSWORD

Remember me

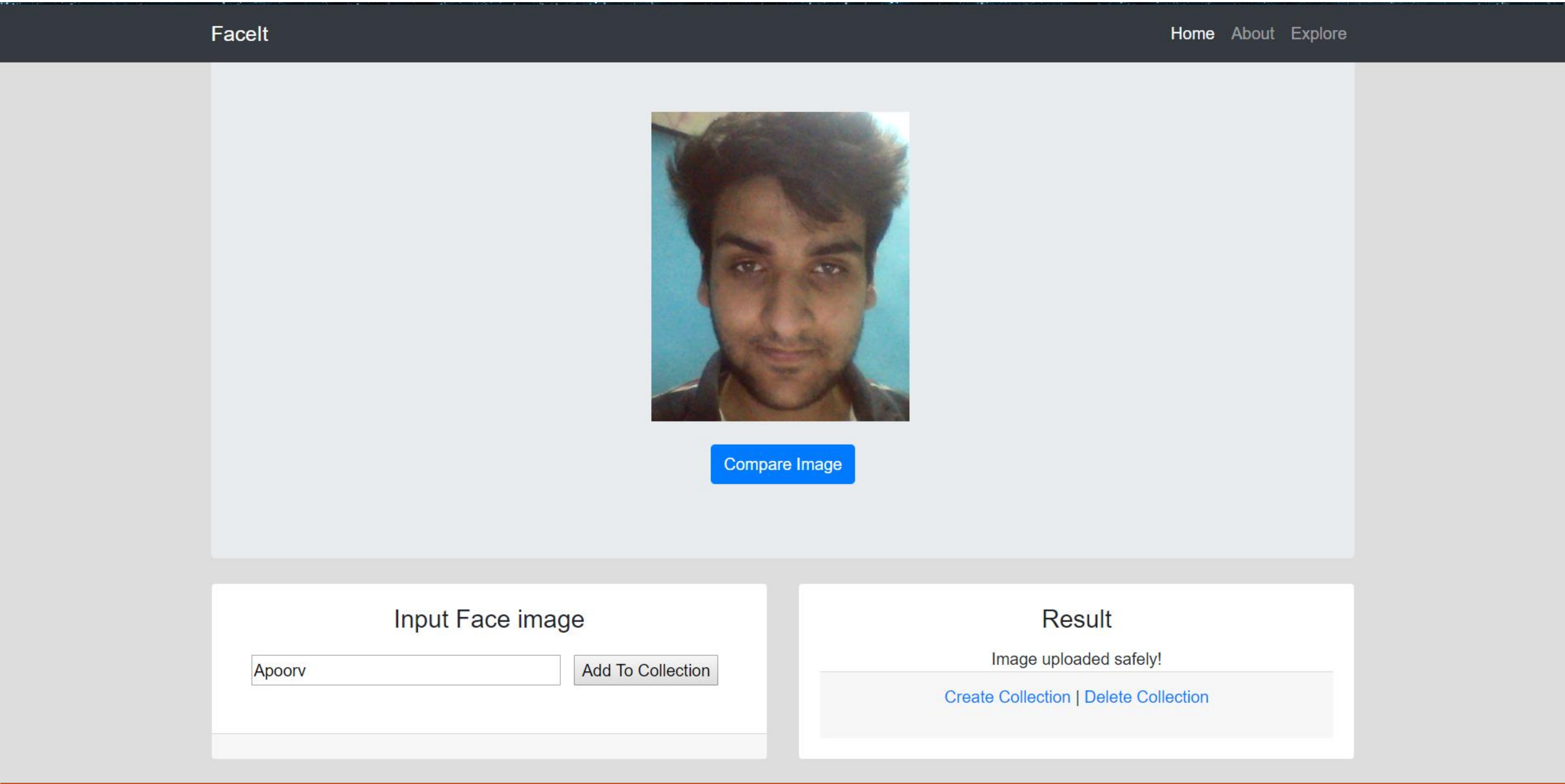
Forgot Password?

Login

## Home Page


localhost:4567/#
FaceIt
Home
About
Explore
Compare Image
Input Face image
Enter name for photo
Add To Collection
Result
Create Collection | Delete Collection

## Uploading Face image Real Time



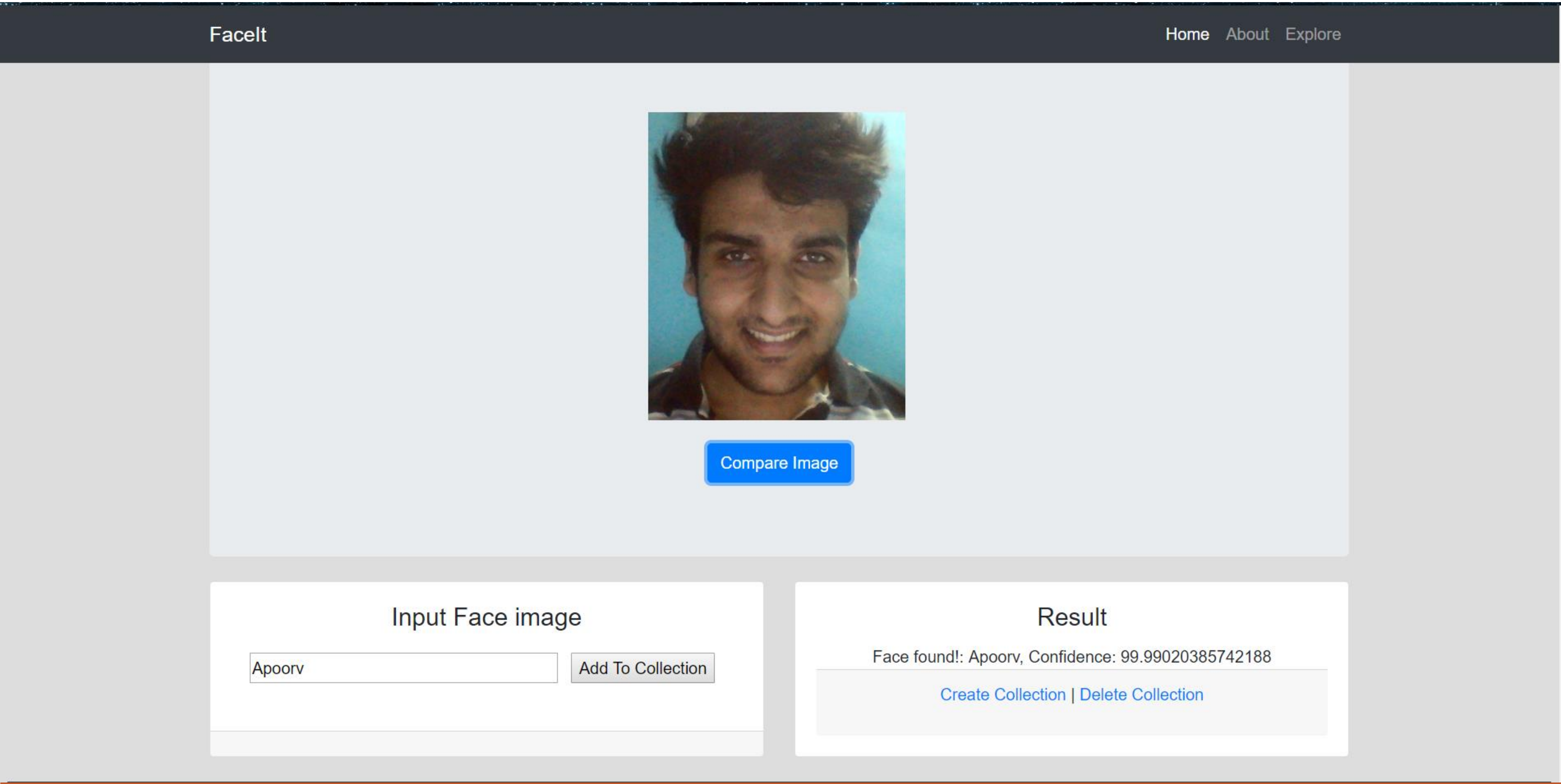
Input Face image

Apoorv

Result

Image uploaded safely!

## Comparing Face Image



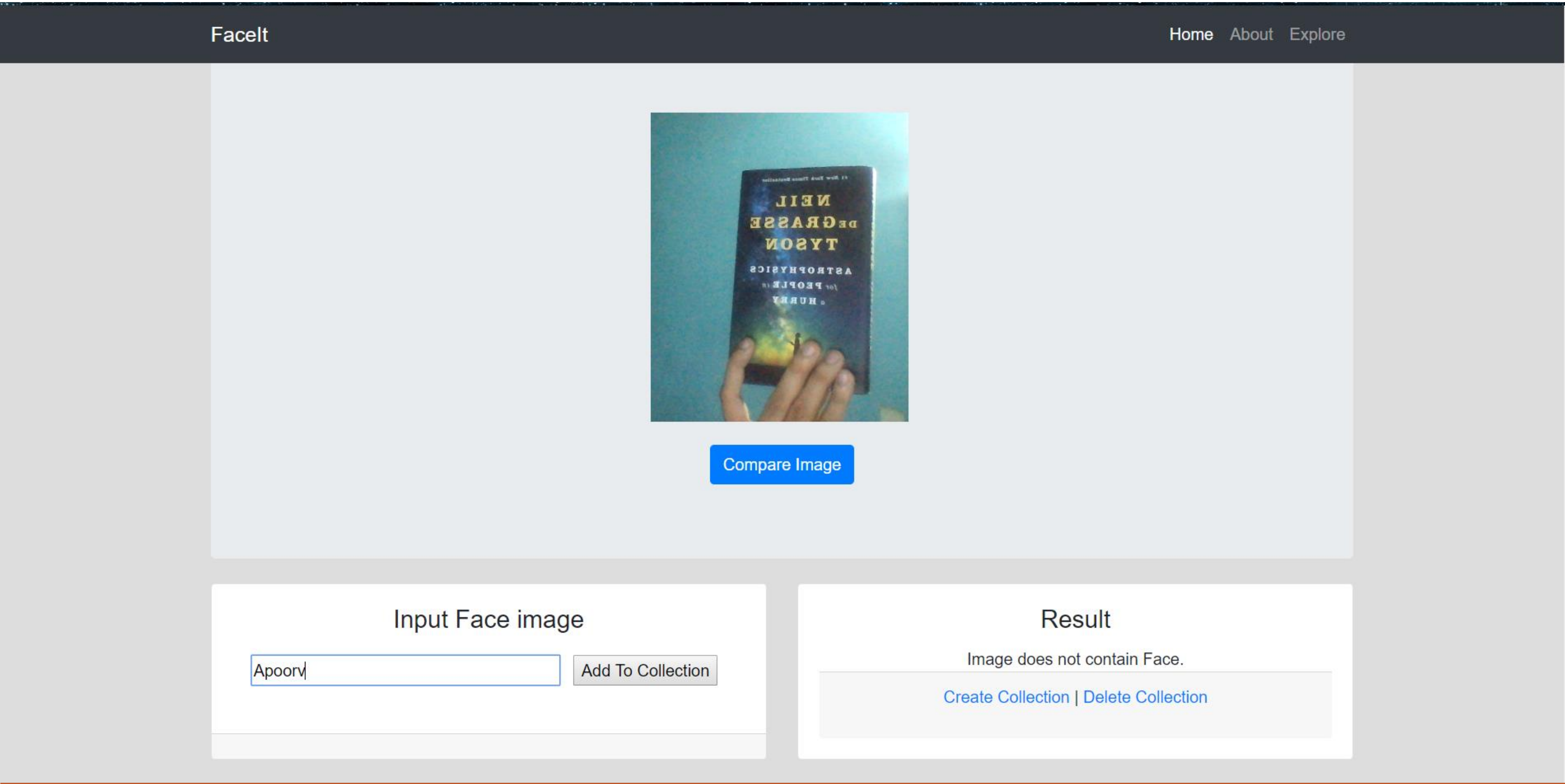
Input Face image

Apoorv

Result

Face found!: Apoorv, Confidence: 99.99020385742188

## Negative Face Detection



Input Face image

Apoorv

Result

Image does not contain Face.

## What needs to be Done

- User Registration/ Login HTML page needs to integrated with the platform.
- Integration of Database Schema with the platform.
- Face-based search functionality needs to be incorporated with the platform.

Thank you