

# CPSC 471 Intermediate Progress Report 1 - Group 43

**Student Name: Igor Zayarny**

**UCID: 10110658**

**Tutorial Number: T08**

**Student Name: Luyi Wang**

**UCID: 30033104**

**Tutorial Number: T10**

**Student Name: Kaumil Patel**

**UCID: 30088096**

**Tutorial Number: T02**

## ***1. Abstract***

With the impact of the pandemic on the general public, each country or city has taken measures to protect its people. Our project is a city entrance/exit inspection system to check the health status of people entering and leaving the city and to record user information. The project includes many functions and is used by a variety of people. Firstly, new users can create their own account and choose their identity. After logging in again, the system will assign them to different screens. Secondly, people who are travelers can enter their personal information and report their health status, which is recorded in the system. The information will be collected and the traveler will be informed whether he/she can enter the city or not. In addition, customs officers are recorded in the system and are divided into three positions: security officer, customs officer and management. Each position has its own tasks related to the management of the travelers, so that the security of the city can be further guaranteed.

## ***2. Describe the problem or task your database was designed to address***

In our project, the database is an integral part. The database enables data transmission and storage, and allows users to receive the information in a timely manner.

First of all, the database is able to store data in the backend. For instance, when Visitors upload their information, health status, license plate and destination location, the database stores that data and makes it accessible to entity types such as border officers, managers, and security guards. Storing and organizing the data in such a way makes it easier to find necessary information.

Secondly, the advantages of using a database include the possibility of data sharing and real-time transfer. The database contains all data that can be accessed by users in a specified interface, and users can communicate with each other by transferring the data in a variety of ways. For example, visitors can enter their personal information and health status in the user interface, the border staff will receive the data, process it, and return an approved/denied status to the visitor's page. Management can also transmit information about changes to personnel data to all employees, ensuring that records are always up-to-date.

In addition, the database can ensure the security and reliability of the data, prevent the loss of data, minimizing the possibility of intrusion, while also protecting the validity and compatibility of the data. Furthermore, the database can prevent data duplication, and can

guarantee the independence and consistency of data. Without the use of the database, the data may be in a scattered state and connections would be difficult to follow and relate to each entity, making searching through and managing the database more difficult. The database ensures the centralized management of data, and also maintains the relationships between data and users through various models. Through all these factors, the user experience of those who interact with and manage the database is improved.

**3. *Describe the system you have created to address the problem or task.***

The system we've created allows different types of users to access and manage data that they are responsible for. Visitors will be able to input their personal information and health status. Border officers will be able to review this data and return an approved/denied status to the visitor. Managers can change the details of employees and the border officer could make decisions of visitor's status (approved or denied), security guards would record the number of cases they have handled.

**4. *A project design section where you discuss the different users of your system, provide a complete picture of the functionality offered by your API.***

There will be four different types of end users. The first is a visitor to the city which would have the lowest privileges. A visitor is able to manage their own account and modify their personal details. The visitor will also be able to see the safety level of locations around the city and the amount of people visiting the location throughout the week. The three end users left are different types of employees. The border officer's job is to process visitors into the city; they can see their personal data of the visitor and make sure it's up to date. The border guard handles security incidents and keeps track of them. Lastly the manager is like the admin for their respective branches and is able to see how his employees are doing.

**5. *The project design section must also include a thorough ER diagram.***

The EER diagram we designed contains 12 entities with 2 weak entities during the design process. There are relationships between these entities and each entity has their own attributes around them. The EER diagram which we designed is shown below.(figure 1). In the EER diagram, there are four types of end-users(Person, Account, Entry/Exit point and Location). Because these entities have key attributes to represent themselves. The system records the information and health status of any person, Visitors are processed by the Border officers and the Manager will take control of employees. The system allows the users to upload and collect information they need to protect the city from the virus. All those functions aim at providing a more convenient user experience.



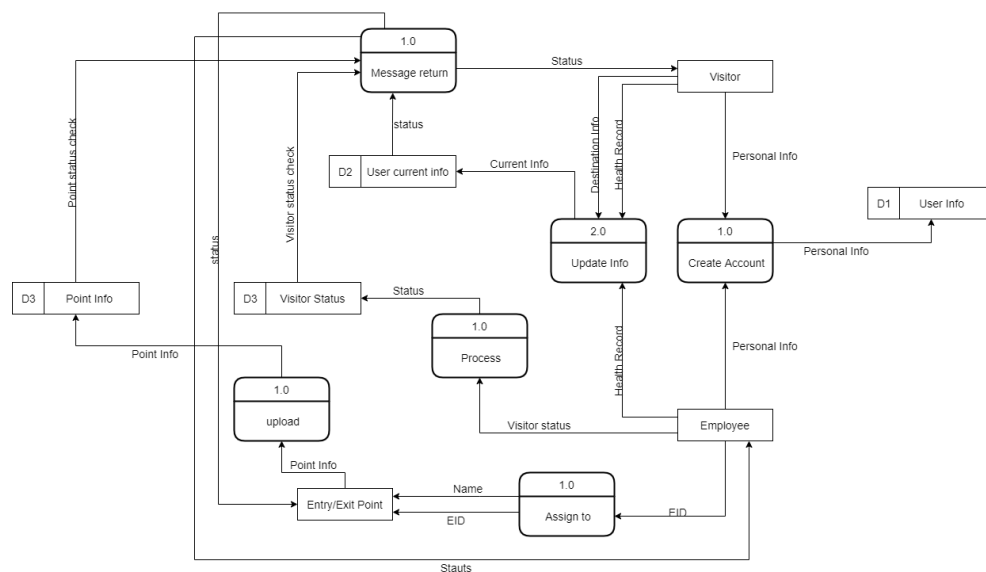
The relational diagram is based on the previous EER diagram above. During our group's meeting and situation, we made some changes on the relational diagram.

- The attribute called Address was aiming for entry/exit point location. We have changed it to describe the destination location of visitors.
- Added attributes called processed and status into entity Visitor. There is a relationship between border officer and visitor which is the approved/denied status.

We add these attributes because the Visitors could upload their destination location into the system and the border officer also could give approved/denied status of visitors based on the health status. This implementation requires the transfermission of data between two different users.

**7. The implementation section should then describe the DBMS you selected for the implementation of the project and must also include the SQL statements for each of the transactions implemented. It is not necessary to discuss these transactions in relational algebra or calculus.**

We used MySQL to create our back-end which is the database. There are several diagrams we create during the implementation. The Hierarchy Input Process Output Diagram, the Data Flow Diagrams and API Functional Tables. All of that work would help us to analyze this database structure and be helpful with the back-end.



**Function:** Add Person

**Input:** @ID, @Name, @Phone, @Address, @Username

**Outputs:** None

**Pseudocode:**

Connect to the database

Query = INSERT INTO Person (@ID, @Name, @Phone, @Address, @Username)

Parse Query

Execute Query

Close Connection to the database

**Function:** Search/View Person

**Input:** @ID

**Outputs:** @ID, @Name, @Phone, @Address, @Username

**Pseudocode:**

Connect to the database

```
Query =      SELECT *
              FROM Person
              WHERE ID = @ID
```

Parse Query

Execute Query

Close Connection to the database

**Function:** Edit Visitor

**Input:** @ID, @Home\_County, @Entry\_Point, @Exit\_Point, @Entry\_Date, @Exit\_Date,  
@Processed\_By, @Status

**Outputs:** None

**Pseudocode:**

Connect to the database

```
Query =      UPDATE Person
              SET Home_County = @Home_County AND Entry_Point = @Entry_Point AND
              Exit_Point = @Exit_Point AND Entry_Date = @Entry_Date AND Exit_Date =
              @Exit_Date AND Processed_By = @Processed_By AND Status = @Status
              WHERE (ID = @ID)
```

Parse Query

Execute Query

Close Connection to the database

**Function:** Delete Account

**Input:** @Username, @Password

**Outputs:** None

**Pseudocode:**

Connect to the database

```
Query =      DELETE FROM Account WHERE Username = @Username AND Password =
              @Password
```

Parse Query

Execute Query

Close Connection to the database

**Function:** Add Employee

**Input:** @EID, @ID, @Start\_Date

**Outputs:** None

**Pseudocode:**

Connect to the database

```
Query =      INSERT INTO Employee (@EID, @ID, @Start_Date)
```

Parse Query

Execute Query

Close Connection to the database

DROP DATABASE IF EXISTS BorderService;

CREATE DATABASE BorderService;

USE BorderService;

```
CREATE TABLE `account` (  
  username          VARCHAR(16) NOT NULL,  
  `password`        VARCHAR(45) NOT NULL,  
  `privilege`        VARCHAR(45) DEFAULT NULL,  
  PRIMARY KEY (username)  
);
```

```
CREATE TABLE person (  
  ID                INT NOT NULL,  
  `name`            VARCHAR(45) NOT NULL,  
  phone             VARCHAR(45) NOT NULL,  
  address            VARCHAR(45) NOT NULL,  
  username           VARCHAR(45) NOT NULL,  
  PRIMARY KEY (ID),  
  KEY username_idx (username),  
  CONSTRAINT username FOREIGN KEY (username) REFERENCES `account` (username) ON  
UPDATE CASCADE  
);
```

```
CREATE TABLE branch (  
  branchID          INT NOT NULL,  
  accesstype         VARCHAR(45) NOT NULL,  
  borderingcountry   VARCHAR(45) NOT NULL,  
  address            VARCHAR(45) NOT NULL,  
  PRIMARY KEY (branchID)  
);
```

```
CREATE TABLE employee (  
  EID               INT NOT NULL,  
  ID                INT NOT NULL,  
  startdate         DATE NOT NULL,  
  PRIMARY KEY (EID),  
  KEY ID (ID),  
  CONSTRAINT empID FOREIGN KEY (ID) REFERENCES person (ID) ON UPDATE  
CASCADE  
);
```

```
CREATE TABLE borderofficer (  
  EID               INT NOT NULL,  
  numapproved       INT NOT NULL,  
  numdenied          INT NOT NULL,  
  branchID          INT NOT NULL,  
  PRIMARY KEY (EID),  
  CONSTRAINT branchID FOREIGN KEY (branchID) REFERENCES branch (branchID) ON  
UPDATE CASCADE,  
  CONSTRAINT EID FOREIGN KEY (EID) REFERENCES employee(EID) ON UPDATE  
CASCADE  
);
```

```

CREATE TABLE visitor (
  ID INT NOT NULL,
  homecountry VARCHAR(45) NOT NULL,
  entrypoint INT DEFAULT NULL,
  exitpoint INT DEFAULT NULL,
  entrydate DATE DEFAULT NULL,
  exitdate DATE DEFAULT NULL,
  processedby INT DEFAULT NULL,
  `status` VARCHAR(45) DEFAULT NULL,
  PRIMARY KEY (ID),
  KEY entrypoint_idx (entrypoint),
  KEY exitpoint_idx (exitpoint),
  KEY processedby_idx (processedby),
  CONSTRAINT entrypoint FOREIGN KEY (entrypoint) REFERENCES branch (branchID) ON
UPDATE CASCADE,
  CONSTRAINT exitpoint FOREIGN KEY (exitpoint) REFERENCES branch (branchID) ON
UPDATE CASCADE,
  CONSTRAINT ID FOREIGN KEY (ID) REFERENCES person (ID) ON UPDATE CASCADE,
  CONSTRAINT processedby FOREIGN KEY (processedby) REFERENCES borderofficer (EID)
ON UPDATE CASCADE
);

```

```

CREATE TABLE dependent (
  visitorID INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (visitorID, name),
  CONSTRAINT visitorID FOREIGN KEY (visitorID) REFERENCES person (ID) ON UPDATE
CASCADE
);

```

```

CREATE TABLE healthrecord (
  personID INT NOT NULL,
  gender VARCHAR(45) NOT NULL,
  lasttestdate DATE DEFAULT NULL,
  PRIMARY KEY (personID),
  CONSTRAINT personID FOREIGN KEY (personID) REFERENCES person (ID) ON UPDATE
CASCADE
);

```

```

CREATE TABLE vehicle (
  vvisitorID INT NOT NULL,
  licensenum VARCHAR(45) NOT NULL,
  PRIMARY KEY (vvisitorID),
  CONSTRAINT vvisitorID FOREIGN KEY (vvisitorID) REFERENCES visitor (ID) ON UPDATE
CASCADE
);

```

```

CREATE TABLE securityguard (
  sgEID INT NOT NULL,
  numincidents INT NOT NULL,

```

```

    sgbranchID          INT NOT NULL,
    PRIMARY KEY (sgEID),
    KEY branchID_idx (sgbranchID),
    CONSTRAINT sgbranchID FOREIGN KEY (sgbranchID) REFERENCES branch (branchID) ON
UPDATE CASCADE,
    CONSTRAINT sgEID FOREIGN KEY (sgEID) REFERENCES employee (EID) ON UPDATE
CASCADE
);

```

```

CREATE TABLE manager (
    mEID                INT NOT NULL,
    numsupervised       INT DEFAULT NULL,
    mbranchID          INT NOT NULL,
    PRIMARY KEY (mEID),
    KEY mbranchID_idx (mbranchID),
    CONSTRAINT mbranchID FOREIGN KEY (mbranchID) REFERENCES branch (branchID) ON
UPDATE CASCADE,
    CONSTRAINT mEID FOREIGN KEY (mEID) REFERENCES employee (EID) ON UPDATE
CASCADE
);

```

```

CREATE TABLE location (
    address             VARCHAR(45) NOT NULL,
    `name`              VARCHAR(45) DEFAULT NULL,
    safetylevel         INT NOT NULL,
    PRIMARY KEY (address)
);

```

```

CREATE TABLE visits (
    visitaddress        VARCHAR(45) NOT NULL,
    visitvisitorID      INT NOT NULL,
    visitDate           DATE NOT NULL,
    duration            INT DEFAULT NULL,
    KEY visitorID_idx (visitvisitorID),
    CONSTRAINT visitaddress FOREIGN KEY (visitaddress) REFERENCES location (address) ON
UPDATE CASCADE,
    CONSTRAINT visitvisitorID FOREIGN KEY (visitvisitorID) REFERENCES visitor (ID) ON
UPDATE CASCADE
);

```

```

INSERT INTO account (username, password) VALUES
('Tom', '123', '0'),
('Hank', '123', '1'),
('Bob', '456', '2'),
('James', '456', '3'),
('David', '789', '0');

```

```

INSERT INTO person (ID, `name`, phone, address, username) VALUES
(1, 'Hank H', '1234567890', '11 city ave, calgary', 'Hank'),
(2, 'Bob B', '1111111111', '22 country ave, calgary', 'Bob'),
(3, 'Tom T', '2222222222', '33 town ave, calgary', 'Tom'),

```



(4, 'James B', '1112223333', '44 country street, calgary', 'James'),  
(5, 'David T', '4445556666', '33 town street, calgary', 'David');

INSERT INTO branch(branchID, accesstype, borderingcountry, address) VALUES  
(11, 'water', 'Canada', 'City Border, Vancouver'),  
(22, 'land', 'US', 'Town Border, Vancouver');

INSERT INTO employee(EID, ID, startdate) VALUES  
(1234, 3, '2021-01-01'),  
(1235, 4, '2021-01-02'),  
(1236, 5, '2021-01-03');

INSERT INTO borderofficer(EID, numapproved, numdenied, branchID) VALUES  
(1234, 4, 2, '11');

INSERT INTO securityguard(sgEID, numincidents, sgbranchID) VALUES  
(1235, 10, '22');

INSERT INTO manager(mEID, numsupervised, mbranchID) VALUES  
(1236, 1, '11');

INSERT INTO location(address, `name`, safetylevel) VALUES  
(440 Historic Ave', 'city park', 1),  
(1 Beach lane', 'Sunshine Pier', 3),  
(111 City Park', 'Central Plaza', 5);

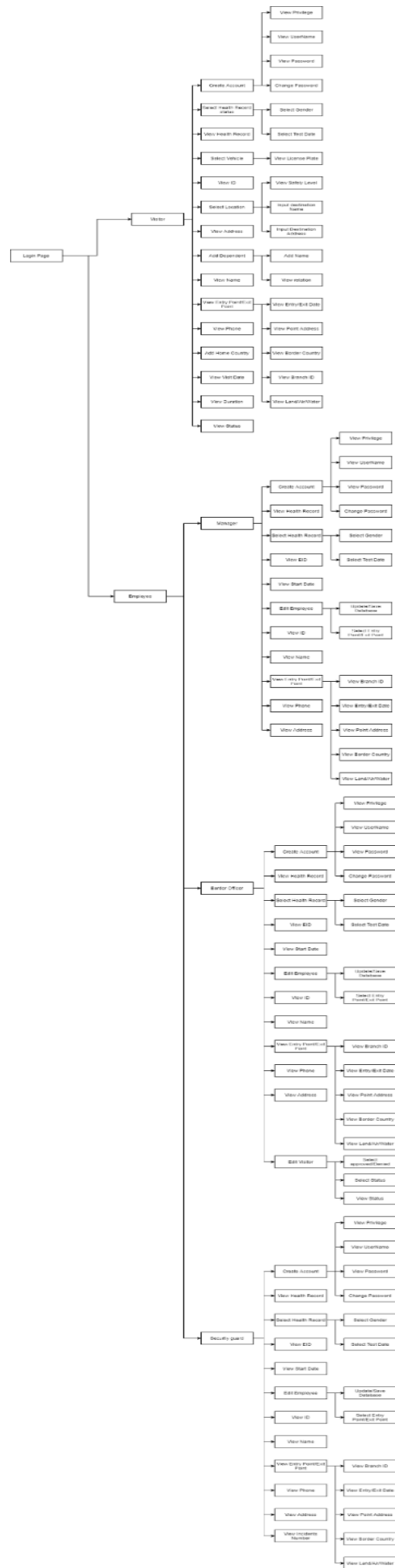
INSERT INTO visitor (ID, homecountry, entrypoint, exitpoint, entrydate, exitdate, processedby, `status`) VALUES  
(1,'Canada', 11, null, '2021-01-11', null, 1234, 'Y'),  
(2,'US', 22, null, '2021-01-12', null, 1234, 'Y');

INSERT INTO visits (visitaddress, visitvisitorID, visitDate, duration) VALUES  
(440 Historic Ave', 1, '2021-02-01', 5),  
(1 Beach Lane', 1, '2021-04-01', 5),  
(440 Historic Ave', 2, '2021-02-01', 3);

INSERT INTO healthrecord (personID, gender, lasttestdate) VALUES  
(1, 'male', '2020-12-12'),  
(2, 'female', '2020-12-17');

INSERT INTO dependent (visitorID, `name`) VALUES  
(1, 'Tommy');

INSERT INTO vehicle (vvisitorID, licensenum) VALUES  
(2, '16848919'),  
(1, '58269518');

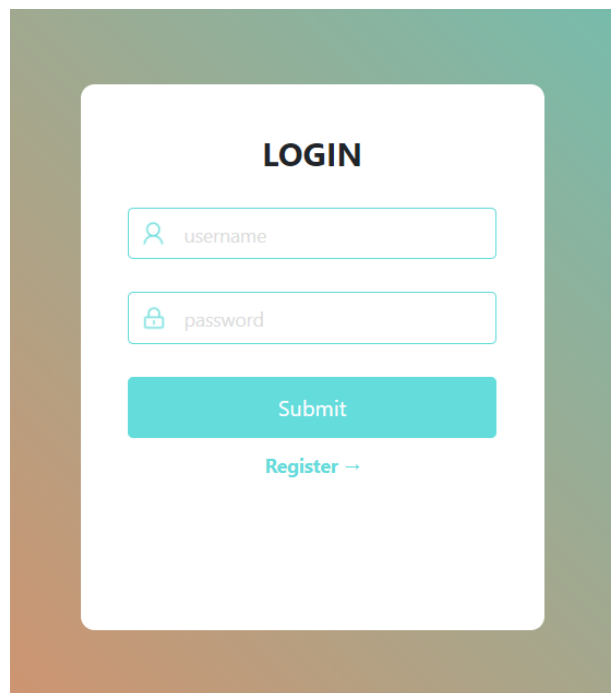


8. *A professional API documentation using the postman tool. Also, you have to use the built-in API documentation feature provided by Postman.*

<https://documenter.getpostman.com/view/15372747/TzJpgzKs>

9. *A quick user guide for people other than you and your TA to be able to smoothly use your project outcome, you must show all functionalities of your Web interface.*

### Login Page

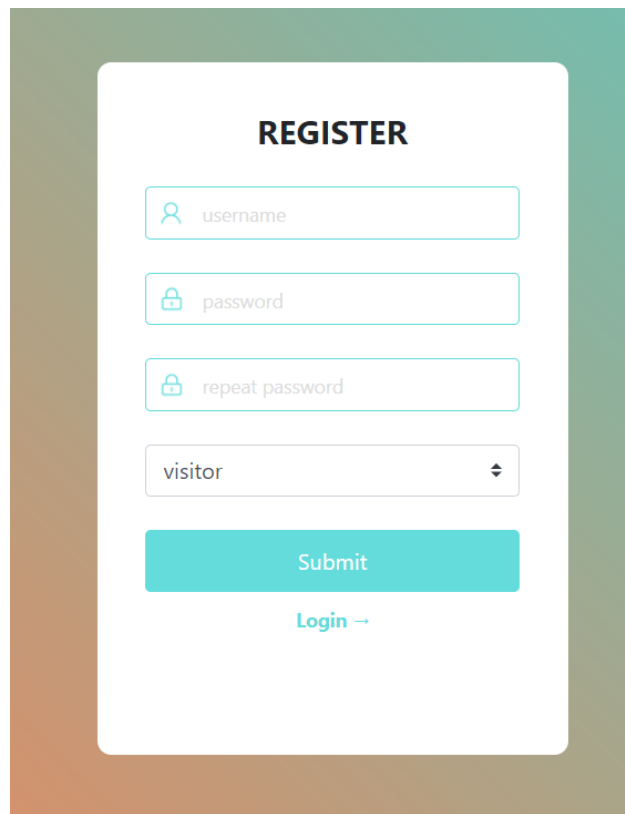


The image shows a login page with a teal and brown gradient background. In the center is a white rounded rectangle containing the following elements:

- The word **LOGIN** in bold black text.
- A username input field with a light blue border and a person icon on the left. The placeholder text is "username".
- A password input field with a light blue border and a lock icon on the left. The placeholder text is "password".
- A teal button with the text "Submit" in white.
- A link that says "Register →" in teal text.

- This is the first screen that the user could access. There are two username and password input bars and two buttons which could submit the info or register an account.
- There are two circumstances in which the error message will pop-up:
  - when the username and the password are not matching with each other.
  - when the username does not exist in the database.
- Everytime the users are logging out from the system, they will end up on this page.

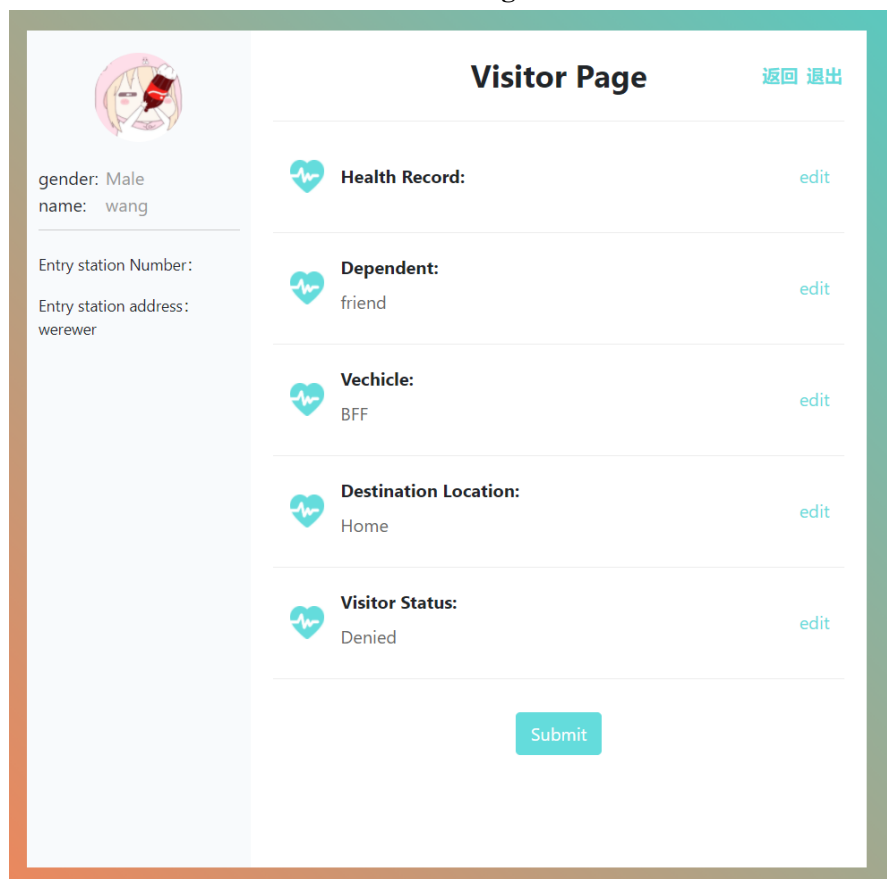
## Register/New Account Page



The image shows a registration form titled "REGISTER" centered on a white background with a light green border. The form contains four input fields: "username" with a person icon, "password" with a lock icon, "repeat password" with a lock icon, and a role selector dropdown currently showing "visitor". Below these fields is a teal "Submit" button and a teal link "Login →".

- This is the register screen. New users could create a new account using a username, password and select their role which is visitor or employee.
- If the password and repeat password are not matching with each other, the error message will pop-up. And when you submit all your information, the submit button would lead you back to the login page.

## Visitor Page

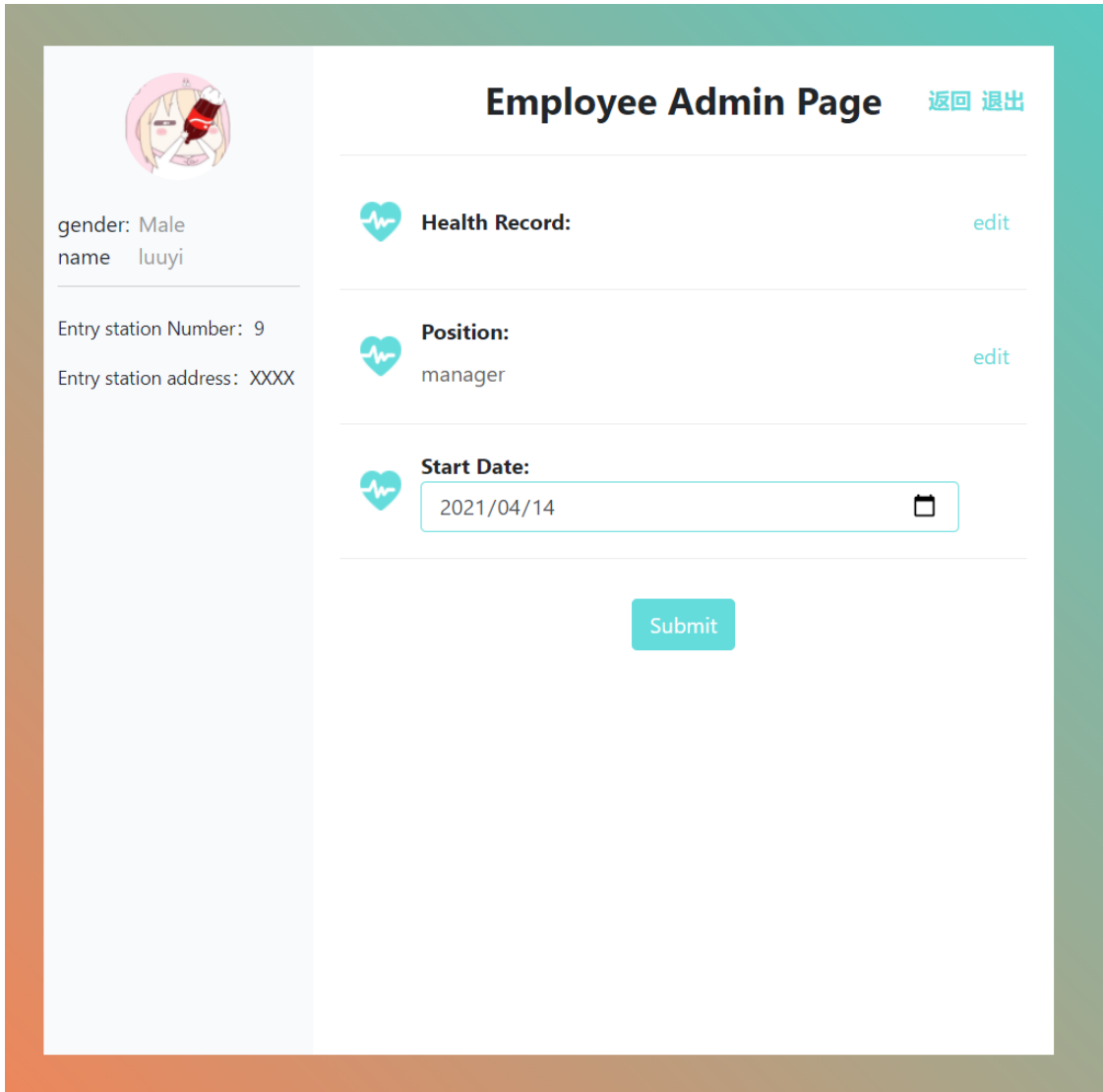



The image shows a "Visitor Page" for a user named Wang. On the left is a profile sidebar with a cartoon avatar, gender "Male", name "wang", and entry station details. The main area lists five categories: "Health Record", "Dependent", "Vechicle" (sic), "Destination Location", and "Visitor Status", each with a value and an "edit" link. A teal "Submit" button is at the bottom.

Field	Value	Action
Health Record		edit
Dependent	friend	edit
Vechicle	BFF	edit
Destination Location	Home	edit
Visitor Status	Denied	edit

- This is the Visitor page. Users could click the edit to the right side to upload their information and the data will be returned to the screen. Including Health record, dependent, vehicle license plate, destination location and visitor status.
- The user information would be on the top left corner and the entry station number and address will also be by the side.
- Visitors could submit the information and wait for the status, it will be updated by border officers after all. And the Visitor could click the button on the top right to go back or log out the system.

### Employee Admin Page






gender: Male  
name luuyi


Entry station Number: 9  
Entry station address: XXXX

## Employee Admin Page


[返回](#) [退出](#)




**Health Record:** [edit](#)



**Position:** [edit](#)  
manager




**Start Date:**  



[Submit](#)

- This is the employee admin Page. Every employee would be entered into this page and they could upload their health record and position information in it, also including the start date of working. The User profile and entry station is still on the top left corner.
- After the employee has updated the position, the submit button will lead him/her into the specific page which matches his position. (Manager, Border Officer and security Guard). And the Visitor could click the button on the top right to go back or log out the system.\

## Manager Page



gender: Male  
name: luuyi

---

Entry station Number: 9  
Entry station address: XXXX

### Manager Page


[返回](#) [退出](#)

xxx	security-guard	<a href="#">Process</a>
xxx2	employee-admin	<a href="#">Process</a>
xxx3	manager	<a href="#">Process</a>
xxx4	border-officer	<a href="#">Process</a>

Submit

- This is the Manager Page. The manager could see all the employee's name and their position, also the user and entry/exit station information at the top left corner. And the user could click the button on the top right to go back or log out the system.
- By pressing the process button on the right side, the manager could transfer any employees to a specific entry/exit station and change their position. The data will be saved in the database and sent to different users.

## Security-guard Page



gender: male  
name: mm

---

Entry station Number: 9  
Entry station address: XXXX

### Event List

[Back](#) [Logout](#)

Incidents handled Number:4

xxx	In progress	<a href="#">Process</a>
xxx2	Solved	<a href="#">Process</a>
xxx3	In progress	<a href="#">Process</a>
xxx4	Solved	<a href="#">Process</a>

Submit

- This is the security-guard page, the security guard could take control of every event and record it. There is an incident handled number at the top of the page which records the number has been handled. And there is user and station information at the top left corner.
- By clicking the process button, the guard could update the progress of any specific event. There are two events, which are In Progress and Solved. After the update of the events, they could click submit to update the number of the incidents handled number at the top. And the user could click the button on the top right to go back or log out the system.

### Border Officer Page

Approved Number:2	Denied Number:2	
xxx	approve	Process
xxx2	denied	Process
xxx	approve	Process
xxx2	denied	Process

Submit

- This is the border Officer Page, which could control the passable status of visitors and record it. There is an approved number and denied number at the top of the page which records the passable number. And there is user and station information at the top left corner.
- By clicking the process button, the officers could view the visitors information and update their status based on their health record. So the current status of the selected visitor will show on the table, and the status will be sent back to the visitor's page after the officers press the submit button. And the user could click the button on the top right to go back or log out the system.

### Stored Procedure Samples

```
CREATE DEFINER='root'@'localhost' PROCEDURE `BOGetBranchID`(IN $EID INT)
BEGIN
    SELECT branchID FROM borderofficer WHERE EID = $EID;
END
```

```
CREATE DEFINER='root'@'localhost' PROCEDURE `GetDependents`(IN VID INT)
BEGIN
    SELECT name FROM dependent WHERE visitorID = VID;
END
```

```
CREATE DEFINER='root'@'localhost' PROCEDURE `GetLastTestDate`(IN VID INT)
BEGIN
    SELECT lasttestdate FROM healthrecord WHERE personID = VID;
END
```

```
CREATE DEFINER='root'@'localhost' PROCEDURE `GetVisitorsByLocation`(IN VAddress
VARCHAR(45))
BEGIN
    SELECT visitvisitorID FROM visits WHERE visitaddress = VAddress;
END
```