CSI2132 Deliverable 2 – Group #188

Group Members: Mahmoud Fawaz 300162088, Sam Pich 300290697, Victor Nguyen 300283734

GitHub link: https://github.com/mfawaz092/e-Hotels-Project

Video link: https://drive.google.com/file/d/1Astia0l9v6Cmg7_GqtJ-kMCDo5FHtu5m/view?usp=sharing

- **DBMS:** PostgreSQL
- **Programming languages:**
  1) Front-end: HTML, CSS, JavaScript
  2) Back-end: Java

- **To run our web app:**
  1) Open the project in IntelliJ IDE.
  2) Run Apache Tomcat and access our webapp through the local server provided by Tomcat.

- **DDLs:**

1) Tables:

```
CREATE TABLE HotelChain (
    chain_id SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    numberOfHotels INTEGER NOT NULL
);

CREATE TABLE HotelChain_ADDRESS (
    chain_id INTEGER REFERENCES HotelChain(chain_id) ON DELETE CASCADE,
    address VARCHAR(255) NOT NULL
```

```
);

CREATE TABLE HotelChain_EMAIL (
    chain_id INTEGER REFERENCES HotelChain(chain_id) ON DELETE CASCADE,
    email VARCHAR(255) NOT NULL
);

CREATE TABLE HotelChain_PHONE (
    chain_id INTEGER REFERENCES HotelChain(chain_id) ON DELETE CASCADE,
    phone VARCHAR(20) NOT NULL
);

CREATE TABLE Hotel (
    hotel_id SERIAL PRIMARY KEY,
    category VARCHAR(50) NOT NULL,
    name VARCHAR(255) NOT NULL,
    address VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    numberOfRooms INTEGER NOT NULL,
    stars INTEGER NOT NULL,
    chain_id INTEGER REFERENCES HotelChain(chain_id) ON DELETE CASCADE
);

CREATE TABLE Hotel_PHONE (
    hotel_id INTEGER REFERENCES Hotel(hotel_id) ON DELETE CASCADE,
    phone VARCHAR(20) NOT NULL
);
```

```
CREATE TABLE Owns (
    chain_id INTEGER REFERENCES HotelChain(chain_id) ON DELETE CASCADE,
    hotel_id INTEGER REFERENCES Hotel(hotel_id) ON DELETE CASCADE,
    PRIMARY KEY (chain_id, hotel_id)
);

CREATE TABLE Manager (
    manager_id SERIAL PRIMARY KEY,
    ssn_sin VARCHAR(20) NOT NULL,
    name VARCHAR(255) NOT NULL,
    address VARCHAR(255) NOT NULL
);

CREATE TABLE Employee (
    employee_id SERIAL PRIMARY KEY,
    ssn_sin VARCHAR(20) NOT NULL,
    name VARCHAR(255) NOT NULL,
    address VARCHAR(255) NOT NULL,
    hotel_id INTEGER REFERENCES Hotel(hotel_id) ON DELETE CASCADE
);

CREATE TABLE Employee_POSITION (
    employee_id INTEGER REFERENCES Employee(employee_id) ON DELETE
CASCADE,
    ssn_sin VARCHAR(20) NOT NULL,
    Position VARCHAR(50) NOT NULL,
```

```
    PRIMARY KEY (employee_id, ssn_sin)
);

CREATE TABLE Employs (
    hotel_id INTEGER REFERENCES Hotel(hotel_id) ON DELETE CASCADE,
    manager_id INTEGER REFERENCES Manager(manager_id) ON DELETE
CASCADE,
    PRIMARY KEY (hotel_id, manager_id)
);

CREATE TABLE Room (
    room_id SERIAL PRIMARY KEY,
    price NUMERIC(10, 2) NOT NULL,
    capacity INTEGER NOT NULL,
    extendable BOOLEAN NOT NULL,
    view VARCHAR(50) NOT NULL,
    hotel_id INTEGER REFERENCES Hotel(hotel_id) ON DELETE CASCADE
);

CREATE TABLE Room_CONDITION (
    room_id INTEGER REFERENCES Room(room_id) ON DELETE CASCADE,
    Condition VARCHAR(255) NOT NULL
);

CREATE TABLE Room_AMENITIES (
    room_id INTEGER REFERENCES Room(room_id) ON DELETE CASCADE,
    Amenities VARCHAR(255) NOT NULL
```

```
);

CREATE TABLE Has (
    hotel_id INTEGER REFERENCES Hotel(hotel_id) ON DELETE CASCADE,
    room_id INTEGER REFERENCES Room(room_id) ON DELETE CASCADE,
    PRIMARY KEY (hotel_id, room_id)
);

CREATE TABLE Customer (
    customer_id SERIAL PRIMARY KEY,
    id_type VARCHAR(50) NOT NULL,
    name VARCHAR(255) NOT NULL,
    address VARCHAR(255) NOT NULL,
    registration_date DATE NOT NULL
);

CREATE TABLE SearchesFor (
    room_id INTEGER REFERENCES Room(room_id) ON DELETE CASCADE,
    customer_id INTEGER REFERENCES Customer(customer_id) ON DELETE
CASCADE,
    id_type VARCHAR(50) NOT NULL,
    PRIMARY KEY (room_id, customer_id, id_type)
);

CREATE TABLE Booking (
    booking_id SERIAL PRIMARY KEY,
    booking_date DATE NOT NULL,
```

```
    check_in_date DATE NOT NULL,
    check_out_date DATE NOT NULL,
    customer_id INTEGER REFERENCES Customer(customer_id) ON DELETE
CASCADE,
    room_id INTEGER REFERENCES Room(room_id) ON DELETE CASCADE,
    CHECK (check_out_date >= check_in_date)
);

CREATE TABLE Books (
    customer_id INTEGER REFERENCES Customer(customer_id) ON DELETE
CASCADE,
    id_type VARCHAR(50) NOT NULL,
    booking_id INTEGER REFERENCES Booking(booking_id) ON DELETE CASCADE,
    PRIMARY KEY (customer_id, id_type, booking_id)
);

CREATE TABLE Renting (
    renting_id SERIAL PRIMARY KEY,
    renting_date DATE NOT NULL,
    booking_id INTEGER REFERENCES Booking(booking_id) ON DELETE CASCADE,
    customer_id INTEGER REFERENCES Customer(customer_id) ON DELETE
CASCADE
);

CREATE TABLE Payment (
    payment_id SERIAL PRIMARY KEY,
    customer_id INTEGER REFERENCES Customer(customer_id) ON DELETE
CASCADE,
```

```
    amount NUMERIC(10, 2) NOT NULL,
    renting_id INTEGER REFERENCES Renting(renting_id) ON DELETE CASCADE,
    payment_method VARCHAR(50) NOT NULL,
    payment_status VARCHAR(50) NOT NULL
);

CREATE TABLE PaysFor (
    payment_id INTEGER REFERENCES Payment(payment_id) ON DELETE
CASCADE,
    customer_id INTEGER REFERENCES Customer(customer_id) ON DELETE
CASCADE,
    id_type VARCHAR(50) NOT NULL,
    renting_id INTEGER REFERENCES Renting(renting_id) ON DELETE CASCADE,
    PRIMARY KEY (payment_id, customer_id, id_type, renting_id)
);
```

2) Populating Hotel Chains:

```
INSERT INTO HotelChain (chain_id, name, numberOfHotels) VALUES
        (1, 'Mariott International', 9),
        (2, 'Four Seasons Hotels and Resorts', 8),
        (3, 'Hilton Worldwide', 10),
        (4, 'Best Western Hotels', 10),
        (5, 'Holiday Inn', 8);
```

3) Populating Hotels:

```
INSERT INTO Hotel (hotel_id, category, name, address, email,
                                numberofrooms, stars, chain_id) VALUES
                                -- Marriott International
(41, 'Standard', 'Marriott Downtown', '123 Yonge St, Toronto, CAN',
'info@marriottdowntown.com', 7, 3, 1),
```

(42, 'Standard', 'Marriott Lakeside', '456 Lakeshore Dr, Vancouver, CAN',
'info@marriottlakeside.com', 8, 3, 1),
(43, 'Economy', 'Marriott Uptown', '789 Elm St, Montreal, CAN',
'info@marriottuptown.com', 9, 2, 1),
(44, 'Economy', 'Marriott Central', '101 Main St, Calgary, CAN',
'info@marriottcentral.com', 10, 2, 1),
(45, 'Luxury', 'Marriott Parkside', '202 Park Ave, New York City, USA',
'info@marriottparkside.com', 11, 5, 1),
(46, 'Luxury', 'Marriott Riverside', '123 Riverside Dr, Los Angeles, USA',
'info@marriottriverside.com', 12, 5, 1),
(47, 'Standard', 'Marriott Oceanview', '456 Ocean Blvd, Miami, USA',
'info@marriottoceanview.com', 8, 4, 1),
(48, 'Standard', 'Marriott Harbourfront', '789 Harbour Dr, Boston, USA',
'info@marriottharbourfront.com', 9, 4, 1),
(49, 'Standard', 'Marriott Seaside', '101 Beach St, San Francisco, USA',
'info@marriottseaside.com', 10, 3, 1),

-- Four Seasons Hotels and Resorts
(50, 'Standard', 'Four Seasons Downtown', '123 Broadway, New York City, USA',
'info@fourseasonsdowntown.com', 8, 3, 2),
(51, 'Standard', 'Four Seasons Lakeside', '456 Lakeview Dr, Los Angeles, USA',
'info@fourseasonslakeside.com', 9, 3, 2),
(52, 'Economy', 'Four Seasons Uptown', '789 Elm St, Chicago, USA',
'info@fourseasonsuuptown.com', 10, 2, 2),
(53, 'Economy', 'Four Seasons Central', '101 Main St, Miami, USA',
'info@fourseasonsucentral.com', 11, 2, 2),
(54, 'Luxury', 'Four Seasons Parkside', '202 Park Ave, San Francisco, USA',
'info@fourseasonsparkside.com', 8, 5, 2),
(55, 'Luxury', 'Four Seasons Riverside', '123 Riverside Dr, Boston, USA',
'info@fourseasonsriverside.com', 9, 5, 2),
(56, 'Standard', 'Four Seasons Oceanview', '456 Ocean Blvd, Seattle, USA',
'info@fourseasonsoceanview.com', 10, 4, 2),
(57, 'Standard', 'Four Seasons Harbourfront', '789 Harbour Dr, Vancouver, CAN',
'info@fourseasonsharbourfront.com', 8, 4, 2),

-- Hilton Worldwide
(58, 'Standard', 'Hilton Downtown', '123 Main St, Toronto, CAN',
'info@hiltondowntown.com', 9, 3, 3),
(59, 'Standard', 'Hilton Lakeside', '456 Lakeview Dr, Vancouver, CAN',
'info@hiltonlakeside.com', 10, 3, 3),

(60, 'Economy', 'Hilton Uptown', '789 Elm St, Montreal, CAN',
'info@hiltonuptown.com', 11, 2, 3),
(61, 'Economy', 'Hilton Central', '101 Main St, Calgary, CAN',
'info@hiltoncentral.com', 12, 2, 3),
(62, 'Luxury', 'Hilton Parkside', '202 Park Ave, New York City, USA',
'info@hiltonparkside.com', 8, 5, 3),
(63, 'Luxury', 'Hilton Riverside', '123 Riverside Dr, Los Angeles, USA',
'info@hiltonriverside.com', 9, 5, 3),
(64, 'Standard', 'Hilton Oceanview', '456 Ocean Blvd, Miami, USA',
'info@hiltonoceanview.com', 10, 4, 3),
(65, 'Standard', 'Hilton Harbourfront', '789 Harbour Dr, Boston, USA',
'info@hiltonharbourfront.com', 8, 4, 3),
(66, 'Standard', 'Hilton Seaside', '101 Beach St, San Francisco, USA',
'info@hiltonseaside.com', 9, 3, 3),
(67, 'Standard', 'Hilton Downtown II', '123 Bay St, Toronto, CAN',
'info@hiltondowntown2.com', 10, 3, 3),

-- Best Western Hotels
(68, 'Standard', 'Best Western Downtown', '123 Yonge St, Toronto, CAN',
'info@bestwesterndowntown.com', 10, 3, 4),
(69, 'Standard', 'Best Western Lakeside', '456 Lakeshore Dr, Vancouver, CAN',
'info@bestwesternlakeside.com', 11, 3, 4),
(70, 'Economy', 'Best Western Uptown', '789 Elm St, Montreal, CAN',
'info@bestwesternuptown.com', 12, 2, 4),
(71, 'Economy', 'Best Western Central', '101 Main St, Calgary, CAN',
'info@bestwesterncentral.com', 8, 2, 4),
(72, 'Luxury', 'Best Western Parkside', '202 Park Ave, New York City, USA',
'info@bestwesternparkside.com', 9, 5, 4),
(73, 'Luxury', 'Best Western Riverside', '123 Riverside Dr, Los Angeles, USA',
'info@bestwesternriverside.com', 10, 5, 4),
(74, 'Standard', 'Best Western Oceanview', '456 Ocean Blvd, Miami, USA',
'info@bestwesternoceanview.com', 8, 4, 4),
(75, 'Standard', 'Best Western Harbourfront', '789 Harbour Dr, Boston, USA',
'info@bestwesternharbourfront.com', 9, 4, 4),
(76, 'Standard', 'Best Western Seaside', '101 Beach St, San Francisco, USA',
'info@bestwesternseaside.com', 10, 3, 4),
(77, 'Standard', 'Best Western Downtown II', '123 King St, Toronto, CAN',
'info@bestwesterndowntown2.com', 10, 3, 4),

```
-- Holiday Inn
(78, 'Standard', 'Holiday Inn Downtown', '123 Dundas St, Toronto, CAN',
'info@holidayinndowntown.com', 6, 3, 5),
(79, 'Standard', 'Holiday Inn Lakeside', '456 Rideau St, Ottawa, CAN',
'info@holidayinnlakeside.com', 7, 3, 5),
(80, 'Economy', 'Holiday Inn Uptown', '789 Elgin St, Montreal, CAN',
'info@holidayinnuptown.com', 8, 2, 5),
(81, 'Economy', 'Holiday Inn Central', '101 Bank St, Calgary, CAN',
'info@holidayinncentral.com', 9, 2, 5),
(82, 'Luxury', 'Holiday Inn Parkside', '202 Stanley St, New York City, USA',
'info@holidayinnparkside.com', 10, 5, 5),
(83, 'Luxury', 'Holiday Inn Riverside', '123 Riverside Dr, Los Angeles, USA',
'info@holidayinnriverside.com', 11, 5, 5),
(84, 'Standard', 'Holiday Inn Oceanview', '456 Ocean Ave, Miami, USA',
'info@holidayinnoceanview.com', 7, 4, 5),
(85, 'Standard', 'Holiday Inn Harbourfront', '789 Harbour Blvd, Boston, USA',
'info@holidayinnharbourfront.com', 8, 4, 5);
```

4) Populating Hotel Rooms:

```
WITH hotel_views AS (
   SELECT
      h.hotel_id,
      ARRAY['City View', 'Park View', 'Lake View', 'River View', 'Street View'] AS
views,
      h.numberofrooms,
      ARRAY_AGG(DISTINCT r.view) AS existing_views
   FROM hotel h
   LEFT JOIN room r ON h.hotel_id = r.hotel_id
   GROUP BY h.hotel_id, h.numberofrooms
)
INSERT INTO room (price, capacity, extendable, view, hotel_id)
SELECT
   CASE
      WHEN h.category = 'Luxury' THEN 500.00 -- Luxury rooms price
      WHEN h.category = 'Standard' THEN 200.00 -- Standard rooms price
      WHEN h.category = 'Economy' THEN 100.00 -- Economy rooms price
   END AS price,
   CASE
```

```
            WHEN h.category = 'Luxury' THEN 2 -- Luxury rooms capacity
            WHEN h.category = 'Standard' THEN 2 -- Standard rooms capacity
            WHEN h.category = 'Economy' THEN 1 -- Economy rooms capacity
        END AS capacity,
        CASE
            WHEN random() < 0.75 THEN TRUE -- 75% of rooms are extendable
            ELSE FALSE
        END AS extendable,
        v.view,
        h.hotel_id
    FROM hotel h
    JOIN hotel_views hv ON h.hotel_id = hv.hotel_id
    JOIN LATERAL unnest(hv.views) v(view) ON true
    JOIN generate_series(1, hv.numberofrooms) AS s ON true;
```

5) Populating Hotel Chain info (phone, address, email):

```
-- Insert addresses for each hotel chain
INSERT INTO hotelchain_address (chain_id, address)
VALUES
    (1, '123 Main St, Toronto, CAN'),
    (2, '456 Lakeview Dr, Vancouver, CAN'),
    (3, '789 Elm St, Montreal, CAN'),
    (4, '101 Park Ave, New York City, USA'),
    (5, '202 Riverside Dr, Los Angeles, USA');


-- Insert email addresses for each hotel chain
INSERT INTO hotelchain_email (chain_id, email)
VALUES
    (1, 'marriott@example.com'),
    (2, 'fourseasons@example.com'),
    (3, 'hilton@example.com'),
    (4, 'bestwestern@example.com'),
```

```
    (5, 'holidayinn@example.com');


-- Insert phone numbers for each hotel chain
INSERT INTO hotelchain_phone (chain_id, phone)
VALUES
    (1, '+1-800-123-4567'), -- Marriott
    (2, '+1-800-234-5678'), -- Four Seasons
    (3, '+1-800-345-6789'), -- Hilton
    (4, '+1-800-456-7890'), -- Best Western
    (5, '+1-800-567-8901'); -- Holiday Inn


-- Insert phone number for every hotel
INSERT INTO public.hotel_phone (hotel_id, phone)
SELECT hotel_id, CONCAT('555-', LPAD(FLOOR(RANDOM() * 900 + 100)::text, 3, '0'),
'-', LPAD(FLOOR(RANDOM() * 9000 + 1000)::text, 4, '0'))
FROM public.hotel;
```

6) Populating customers, manager, and employees:

```
-- Insert 500 random customers
INSERT INTO customer (id_type, name, address, registration_date)
SELECT
    CASE floor(random() * 3)
        WHEN 0 THEN 'Passport'
        WHEN 1 THEN 'Driver License'
        ELSE 'ID Card'
    END AS id_type,
    concat('Customer', customer_id) AS name,
```

```
    concat(floor(random() * 1000) + 1, ' Random St') AS address,
    current_date - interval '365' * floor(random() * 10) AS registration_date
FROM generate_series(1, 500) AS customer_id;

SELECT * FROM public.customer
ORDER BY registration_date ASC

-- Insert managers
INSERT INTO manager (ssn_sin, name, address)
SELECT
    -- Generate a random Social Security Number or Social Insurance Number (SSN/SIN)
    concat(floor(random() * 1000000000), '-', floor(random() * 10000)) AS ssn_sin,
    -- Generate a random name
    concat('Manager_', manager_id) AS name,
    -- Generate a random address
    concat(floor(random() * 1000) + 1, ' Random St') AS address
FROM
    generate_series(1, (SELECT COUNT(*) FROM hotel)) AS manager_id;

--Insert employs relation
INSERT INTO employs (hotel_id, manager_id)
SELECT
    h.hotel_id,
    m.manager_id
FROM
    (SELECT hotel_id, ROW_NUMBER() OVER () AS row_num FROM hotel) AS h
```

```
JOIN
    (SELECT manager_id, ROW_NUMBER() OVER () AS row_num FROM manager) AS
m
ON
    h.row_num = m.row_num;


--Insert employees
INSERT INTO employee (ssn_sin, name, address, hotel_id)
SELECT
    -- Generate a random Social Security Number or Social Insurance Number
(SSN/SIN)
    concat(floor(random() * 1000000000), '-', floor(random() * 10000)) AS ssn_sin,
    -- Generate a random name
    concat('Employee_', employee_id) AS name,
    -- Generate a random address
    concat(floor(random() * 1000) + 1, ' Random St') AS address,
    -- Select a random hotel_id
    sub.hotel_id
FROM
    (SELECT
        generate_series(1, 10) AS employee_id,
        hotel_id
    FROM
        (SELECT hotel_id FROM hotel ORDER BY random() LIMIT (SELECT COUNT(*)
FROM hotel)) AS h
    CROSS JOIN
        generate_series(1, 10) AS s
    ) AS sub;
```

7) Queries:

```
-- Aggregate query. Calculates the average star rating of each hotel chain.
SELECT hc.name AS hotel_chain_name, AVG(h.stars) AS avg_stars_per_chain
FROM Hotel h
JOIN HotelChain hc ON h.chain_id = hc.chain_id
GROUP BY hc.name;

-- Nested query. Retrieves the names of hotels with more than 10 rooms.
SELECT name
FROM hotel
WHERE numberofrooms > 10;

-- Join query. Retrieves names of customers who booked a 'City View' room.
SELECT c.name
FROM customer c
JOIN booking b ON c.customer_id = b.customer_id
JOIN room r ON b.room_id = r.room_id
WHERE r.view = 'City View';

-- Nested query. Retrieves names of hotels with the maximum number of rooms.
SELECT name
FROM hotel
WHERE numberofrooms = (
    SELECT MAX(numberofrooms)
    FROM hotel
);
```

8) Indexes:

-- Frequent queries and data updates we expect:

-- Based on the following description for the web app:
-- "These criteria should be: the dates (start, end) of booking or renting,
-- the room capacity, the area, the hotel chain, the category of the hotel,
-- the total number of rooms in the hotel, the price of the rooms. The user should
-- be able to see the available choices when he/she changes the value of
-- any of these criteria."

-- 1) Indexing hotel chain ID:
CREATE INDEX index_hotel_chain ON Hotel (chain_id);

-- 2) Indexing hotel category:
CREATE INDEX index_hotel_category ON Hotel (category);

-- 3) Indexing check in/out dates for bookings and renting dates for rentings:
CREATE INDEX index_booking_dates ON Booking (check_in_date, check_out_date);
CREATE INDEX index_renting_dates ON Renting (renting_date);

-- 4) Indexing room capacity:
CREATE INDEX index_room_capacity ON Room (capacity);

-- 5) Indexing total number of rooms in the hotel:
CREATE INDEX index_hotel_number_of_rooms ON Hotel (numberOfRooms);

-- 6) Indexing price of the rooms:

```
CREATE INDEX index_room_price ON Room (price);


-- 7) Indexing hotel names:
CREATE INDEX index_hotel_name ON Hotel (name);
    9)  Triggers:
-- Trigger 1: Prevent hotel deletion if has associated rooms
CREATE OR REPLACE FUNCTION prevent_deletion()
RETURNS TRIGGER AS $$
BEGIN
   IF EXISTS (SELECT 1 FROM room WHERE hotel_id = OLD.hotel_id) THEN
      RAISE EXCEPTION 'Cannot delete hotel.';
   END IF;
   RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_prevent_deletion
BEFORE DELETE ON hotel
FOR EACH ROW
EXECUTE FUNCTION prevent_deletion();

-- Trigger 2: Update amt of hotels after insertion or deletion
CREATE OR REPLACE FUNCTION update_hotel_amount()
RETURNS TRIGGER AS $$
BEGIN
   IF TG_OP = 'INSERT' THEN
      UPDATE hotelchain
```

```
        SET numberofhotels = numberofhotels + 1
        WHERE chain_id = NEW.chain_id;
    ELSIF TG_OP = 'DELETE' THEN
        UPDATE hotelchain
        SET numberofhotels = numberofhotels - 1
        WHERE chain_id = OLD.chain_id;
    END IF;
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_update_hotel_amount
AFTER INSERT OR DELETE ON hotel
FOR EACH ROW
EXECUTE FUNCTION update_hotel_amount();
```

10) Views:

```
-- View 1: Availible rooms per area
CREATE OR REPLACE VIEW view_available_rooms AS
SELECT
    h.address AS area,
    COUNT(*) AS available_rooms
FROM
    room r
    JOIN hotel h ON r.hotel_id = h.hotel_id
WHERE
    r.room_id NOT IN (
```

```
        SELECT room_id
        FROM booking
        WHERE check_in_date <= CURRENT_DATE AND check_out_date >=
CURRENT_DATE
    )
GROUP BY
    h.address;


-- View 2: Capacity of all rooms in hotel
CREATE OR REPLACE VIEW viewroom_capacity AS
SELECT
    h.hotel_id,
    h.name AS hotel_name,
    SUM(r.capacity) AS total_capacity
FROM
    hotel h
    JOIN room r ON h.hotel_id = r.hotel_id
GROUP BY
    h.hotel_id, h.name;
```

Requirement	Start Timestamp
1	0:05
2	0:30
3	1:30
4	2:53
5	6:15
6	7:28
7	9:05
8	9:48
9	Only front-end complete

Table 1. Contents of the video