CS2102 Project Report

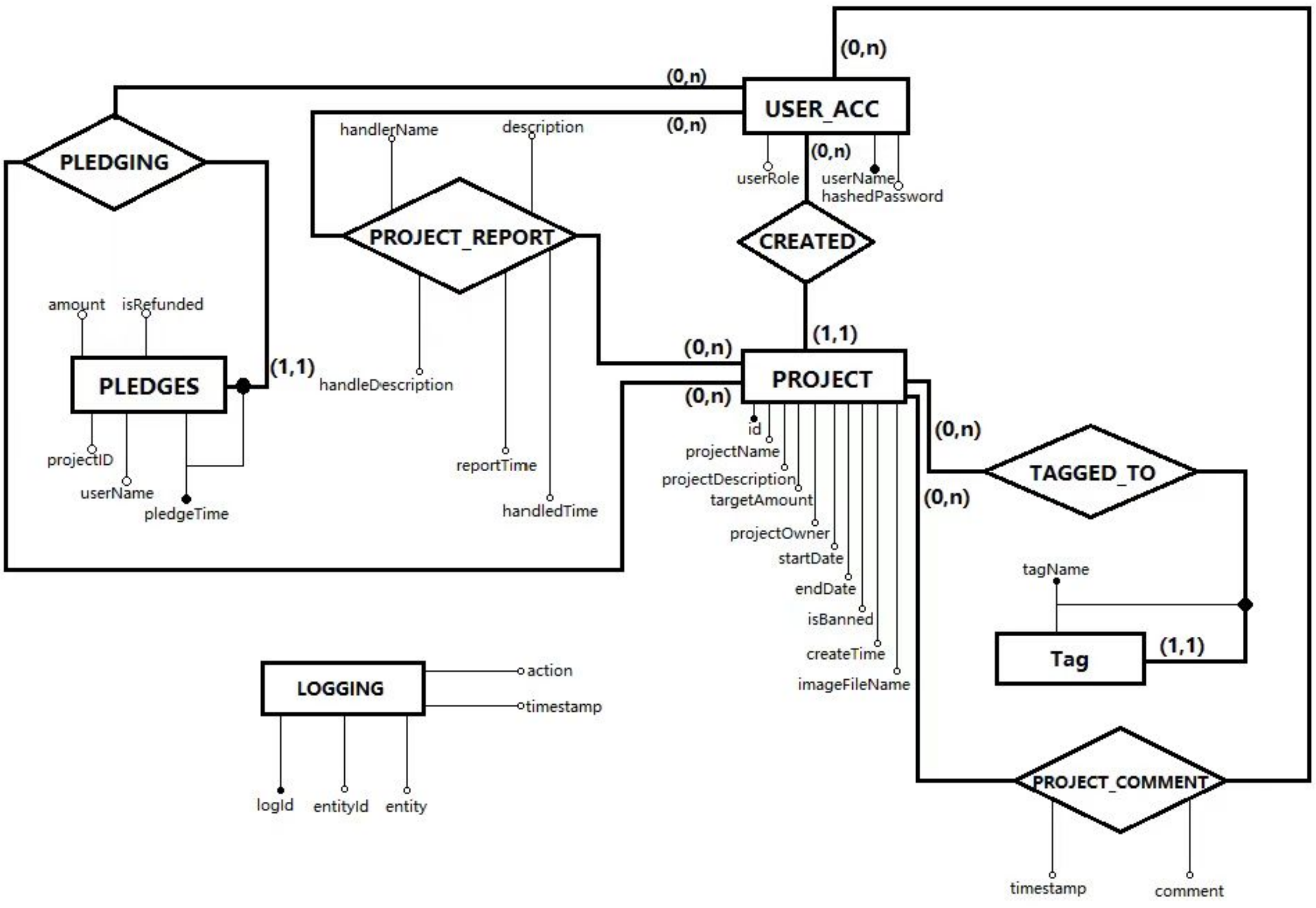
Group 14
Crowdfunding

Jiang Qinhua A0168656M
Wang Shining A0177386M
Yang Luhua A0147986H
Zhang Bosen A0139361E

1. Project information:

In this project, Angular 6 is the frontend framework and we use Express (Nodejs) as backend server and PostgreSQL as the database.

2. Entity-relationship diagram:



3. The translation of the Entity-relationship diagram into relational schema (in SQL DDL code with integrity constraints), chosen sample and representative SQL code (indicate the SQL code and the function of the service it helps to implement):

```sql
CREATE TABLE USER_ACC
(
    userRole VARCHAR(64) NOT NULL CONSTRAINT ROLE_CHK CHECK(userRole = 'USER' OR userRole = 'ADMIN'), -- User / Admin
    userName VARCHAR(64) PRIMARY KEY CONSTRAINT USERNAME_CHK CHECK(length(userName) > 0),
    hashedPassword VARCHAR(64) NOT NULL
);

CREATE TABLE PROJECT (
    id SERIAL PRIMARY KEY,
    projectName VARCHAR(128) NOT NULL,
    projectDescription VARCHAR(2048) NOT NULL,
    targetAmount INTEGER NOT NULL,
    imageFileName VARCHAR(1024),
    projectOwner VARCHAR(64),
    startDate NUMERIC NOT NULL, -- Cannot edit startDate after project have started (Trigger)
    endDate NUMERIC NOT NULL CONSTRAINT DATE_CHK CHECK (endDate > startDate), -- Cannot close project after endDate
    isBanned BOOLEAN DEFAULT FALSE,
    createTime NUMERIC NOT NULL,
    FOREIGN KEY (projectOwner) REFERENCES USER_ACC(userName)
);

CREATE TABLE PLEDGING_PLEDGES
(
    amount INTEGER CONSTRAINT AMOUNT_CHK CHECK (amount > 0),
    pledgeTime NUMERIC NOT NULL,
    projectId INTEGER,
    userName VARCHAR(64),
    isRefunded BOOLEAN DEFAULT FALSE, -- Refund all is project is banned
    PRIMARY KEY (userName, projectId, pledgeTime),
    FOREIGN KEY (userName) REFERENCES USER_ACC(userName),
    FOREIGN KEY (projectId) REFERENCES PROJECT(id) ON DELETE CASCADE
);
CREATE TABLE PROJECT_REPORT
(
    userName VARCHAR(64) NOT NULL,
    projectId INTEGER NOT NULL,
    description VARCHAR(64) NOT NULL,
    handlerName VARCHAR(64), -- Handler Must be Admin (Trigger)
    handleDescription VARCHAR(512),
    reportTime NUMERIC,
    handledTime NUMERIC,
    PRIMARY KEY (userName, projectId),
    FOREIGN KEY (userName) REFERENCES USER_ACC(userName),
    FOREIGN KEY (projectId) REFERENCES PROJECT(id) ON DELETE CASCADE
);

CREATE TABLE TAGGED_TO_TAG
(
    tagName VARCHAR(64),
    projectId INTEGER,
    PRIMARY KEY (tagName, projectId),
    FOREIGN KEY (projectId) REFERENCES PROJECT(id) ON DELETE CASCADE
);

CREATE TABLE LOGGING(
       logId SERIAL PRIMARY KEY,
    entityId VARCHAR(128) NOT NULL,
    action VARCHAR(64) NOT NULL, -- CREATE/DELETE/UPDATE
    entity VARCHAR(64) NOT NULL,
    timestamp NUMERIC NOT NULL
);
```

```sql
CREATE TABLE PROJECT_COMMENT(
    userName VARCHAR(64) NOT NULL,
    projectId SERIAL,
    timestamp NUMERIC NOT NULL,
    comment VARCHAR(255) NOT NULL,
    PRIMARY KEY (userName, projectId, timestamp),
    FOREIGN KEY (userName) REFERENCES USER_ACC(userName),
    FOREIGN KEY (projectId) REFERENCES PROJECT(Id) ON DELETE CASCADE
);

CREATE OR REPLACE FUNCTION autoRefundProjects ()
RETURNS TRIGGER AS $$
DECLARE
BEGIN
IF NEW.isbanned = TRUE THEN
UPDATE PLEDGING_PLEDGES SET isrefunded = TRUE WHERE projectid = NEW.id;
END IF;
RETURN NEW;
END; $$
LANGUAGE PLPGSQL;

CREATE TRIGGER refundPledges
AFTER UPDATE
ON PROJECT
FOR EACH ROW
EXECUTE PROCEDURE autoRefundProjects();

CREATE OR REPLACE FUNCTION handleProjectCheck ()
RETURNS TRIGGER AS $$
DECLARE adminType VARCHAR(64);
BEGIN

IF NEW.handlerName IS NULL
THEN
RETURN NEW;
END IF;

SELECT userRole INTO adminType FROM USER_ACC WHERE userName = NEW.handlerName;
IF adminType='ADMIN'
THEN
RETURN NEW;
ELSE
RAISE EXCEPTION 'USER NOT ADMIN';
RETURN NULL;
END IF;
END; $$
LANGUAGE PLPGSQL;

CREATE TRIGGER checkReportHandling
BEFORE UPDATE
ON PROJECT_REPORT
FOR EACH ROW
EXECUTE PROCEDURE handleProjectCheck();

CREATE OR REPLACE FUNCTION addUserLogging ()
RETURNS TRIGGER AS $$
DECLARE event VARCHAR(64);

BEGIN
    IF (TG_OP = 'DELETE') THEN
        PERFORM logging(TG_OP, CAST(TG_TABLE_NAME AS VARCHAR(64)), CAST(OLD.username AS VARCHAR(128)));
    ELSE
        PERFORM logging(TG_OP, CAST(TG_TABLE_NAME AS VARCHAR(64)), CAST(NEW.username AS VARCHAR(128)));
    END IF;

    RETURN NULL;
END;
$$ LANGUAGE PLPGSQL;
```

```sql
CREATE OR REPLACE FUNCTION addProjectLogging ()
RETURNS TRIGGER AS $$
DECLARE event VARCHAR(64);

BEGIN
    IF (TG_OP = 'DELETE') THEN
        PERFORM logging(TG_OP, CAST(TG_TABLE_NAME AS VARCHAR(64)), CAST(OLD.id AS VARCHAR(128)));
    ELSE
        PERFORM logging(TG_OP, CAST(TG_TABLE_NAME AS VARCHAR(64)), CAST(NEW.id AS VARCHAR(128)));
    END IF;

    RETURN NULL;
END;
$$ LANGUAGE PLPGSQL;


CREATE OR REPLACE FUNCTION logging(action VARCHAR(64), entity VARCHAR(64), entityId VARCHAR(128))
RETURNS VOID AS $$
BEGIN
        INSERT INTO LOGGING (action, entityId, entity, timestamp)
            VALUES (action, entityId, entity, FLOOR(EXTRACT(epoch from now()) * 1000));
    RETURN;
END;
$$ LANGUAGE PLPGSQL;

CREATE TRIGGER createInsertProjectLog
AFTER INSERT
ON PROJECT
FOR EACH ROW
EXECUTE PROCEDURE addProjectLogging();

CREATE TRIGGER createUpdateProjectLog
AFTER UPDATE
ON PROJECT
FOR EACH ROW
EXECUTE PROCEDURE addProjectLogging();

CREATE TRIGGER createDeleteProjectLog
AFTER DELETE
ON PROJECT
FOR EACH ROW
EXECUTE PROCEDURE addProjectLogging();

CREATE TRIGGER createInsertUserLog
AFTER INSERT
ON USER_ACC
FOR EACH ROW
EXECUTE PROCEDURE addUserLogging();

CREATE TRIGGER createUpdateUserLog
AFTER UPDATE
ON USER_ACC
FOR EACH ROW
EXECUTE PROCEDURE addUserLogging();

CREATE TRIGGER createDeleteUserLog
AFTER DELETE
ON USER_ACC
FOR EACH ROW
EXECUTE PROCEDURE addUserLogging();

CREATE OR REPLACE FUNCTION checkProjectLimit ()
RETURNS TRIGGER AS $$
DECLARE currentAmt INTEGER;
DECLARE projectTargetAmt INTEGER;
BEGIN
SELECT SUM(amount) INTO currentAmt FROM PLEDGING_PLEDGES WHERE projectId = NEW.projectId AND isRefunded = false;
SELECT targetAmount INTO projectTargetAmt FROM PROJECT WHERE id = NEW.projectId;
```

```
IF currentAmt IS NULL
THEN
currentAmt = 0;
END IF;

IF NEW.amount + currentAmt > projectTargetAmt
THEN
RAISE EXCEPTION 'AMT EXCEEDS PROJECT LIMIT';
RETURN NULL;
ELSE
RETURN NEW;
END IF;
END; $$
LANGUAGE PLPGSQL;

CREATE OR REPLACE FUNCTION checkProjectDate ()
RETURNS TRIGGER AS $$
DECLARE projectStartDate NUMERIC;
DECLARE projectEndDate NUMERIC;
BEGIN
SELECT startDate INTO projectStartDate FROM PROJECT WHERE id = NEW.projectId;
SELECT endDate INTO projectEndDate FROM PROJECT WHERE id = NEW.projectId;

IF NEW.pledgeTime > projectEndDate THEN
RAISE EXCEPTION 'PROJECT ENDED';
RETURN NULL;
ELSIF NEW.pledgeTime < projectStartDate THEN
RAISE EXCEPTION 'PROJECT NOT STARTED';
RETURN NULL;
ELSE
RETURN NEW;
END IF;
END; $$
LANGUAGE PLPGSQL;

CREATE TRIGGER checkPledgesAmt
BEFORE INSERT
ON PLEDGING_PLEDGES
FOR EACH ROW
EXECUTE PROCEDURE checkProjectLimit();

CREATE TRIGGER checkPledgesDate
BEFORE INSERT
ON PLEDGING_PLEDGES
FOR EACH ROW
EXECUTE PROCEDURE checkProjectDate();
```

4. Assertions & implementations

(1) Only admins can handle a report - At the time of handling the report (Trigger: checkReportHandling, Function: handleProjectCheck)

```
CREATE TRIGGER checkReportHandling
BEFORE UPDATE
ON PROJECT_REPORT
FOR EACH ROW
EXECUTE PROCEDURE handleProjectCheck();

CREATE OR REPLACE FUNCTION handleProjectCheck ()
RETURNS TRIGGER AS $$
DECLARE adminType VARCHAR(64);
BEGIN

IF NEW.handlerName IS NULL
THEN
RETURN NEW;
END IF;
```

```
SELECT userRole INTO adminType FROM USER_ACC WHERE userName = NEW.handlerName;
IF adminType='ADMIN'
THEN
RETURN NEW;
ELSE
RAISE EXCEPTION 'USER NOT ADMIN';
RETURN NULL;
END IF;
END; $$
LANGUAGE PLPGSQL;
```

(2) Project cannot be pledged over its target amount (Trigger: checkPledgesAmt, Function: CheckProjectLimit)

```
CREATE TRIGGER checkPledgesAmt
BEFORE INSERT
ON PLEDGING_PLEDGES
FOR EACH ROW
EXECUTE PROCEDURE checkProjectLimit();
```

```
CREATE OR REPLACE FUNCTION checkProjectLimit ()
RETURNS TRIGGER AS $$
DECLARE currentAmt INTEGER;
DECLARE projectTargetAmt INTEGER;
BEGIN
SELECT SUM(amount) INTO currentAmt FROM PLEDGING_PLEDGES WHERE projectId = NEW.projectId AND isRefunded = false;
SELECT targetAmount INTO projectTargetAmt FROM PROJECT WHERE id = NEW.projectId;

IF currentAmt IS NULL
THEN
currentAmt = 0;
END IF;

IF NEW.amount + currentAmt > projectTargetAmt
THEN
RAISE EXCEPTION 'AMT EXCEEDS PROJECT LIMIT';
RETURN NULL;
ELSE
RETURN NEW;
END IF;
END; $$
LANGUAGE PLPGSQL;
```

(3) Project can only be pledged when it has started (Trigger: checkPledgesDate, Function: checkProjectDate)

```
CREATE TRIGGER checkPledgesDate
BEFORE INSERT
ON PLEDGING_PLEDGES
FOR EACH ROW
EXECUTE PROCEDURE checkProjectDate();
CREATE OR REPLACE FUNCTION checkProjectDate ()
RETURNS TRIGGER AS $$
DECLARE projectStartDate NUMERIC;
DECLARE projectEndDate NUMERIC;
BEGIN
SELECT startDate INTO projectStartDate FROM PROJECT WHERE id = NEW.projectId;
SELECT endDate INTO projectEndDate FROM PROJECT WHERE id = NEW.projectId;

IF NEW.pledgeTime > projectEndDate THEN
RAISE EXCEPTION 'PROJECT ENDED';
RETURN NULL;
ELSIF NEW.pledgeTime < projectStartDate THEN
RAISE EXCEPTION 'PROJECT NOT STARTED';
RETURN NULL;
ELSE
RETURN NEW;
END IF;
END; $$
LANGUAGE PLPGSQL;
```

5. Convience features & implementations

(1) Logging for project and user (Trigger: - All the logging triggers, Functions: addUserLogging, addProjectLogging, logging)

```
CREATE TRIGGER createInsertProjectLog
AFTER INSERT
ON PROJECT
FOR EACH ROW
EXECUTE PROCEDURE addProjectLogging();

CREATE TRIGGER createUpdateProjectLog
AFTER UPDATE
ON PROJECT
FOR EACH ROW
EXECUTE PROCEDURE addProjectLogging();

CREATE TRIGGER createDeleteProjectLog
AFTER DELETE
ON PROJECT
FOR EACH ROW
EXECUTE PROCEDURE addProjectLogging();

CREATE TRIGGER createInsertUserLog
AFTER INSERT
ON USER_ACC
FOR EACH ROW
EXECUTE PROCEDURE addUserLogging();

CREATE TRIGGER createUpdateUserLog
AFTER UPDATE
ON USER_ACC
FOR EACH ROW
EXECUTE PROCEDURE addUserLogging();

CREATE TRIGGER createDeleteUserLog
AFTER DELETE
ON USER_ACC
FOR EACH ROW
EXECUTE PROCEDURE addUserLogging();

CREATE OR REPLACE FUNCTION addUserLogging ()
RETURNS TRIGGER AS $$
DECLARE event VARCHAR(64);

BEGIN
    IF (TG_OP = 'DELETE') THEN
        PERFORM logging(TG_OP, CAST(TG_TABLE_NAME AS VARCHAR(64)), CAST(OLD.username AS VARCHAR(128)));
    ELSE
        PERFORM logging(TG_OP, CAST(TG_TABLE_NAME AS VARCHAR(64)), CAST(NEW.username AS VARCHAR(128)));
    END IF;

    RETURN NULL;
END;
$$ LANGUAGE PLPGSQL;

CREATE OR REPLACE FUNCTION addProjectLogging ()
RETURNS TRIGGER AS $$
DECLARE event VARCHAR(64);

BEGIN
    IF (TG_OP = 'DELETE') THEN
        PERFORM logging(TG_OP, CAST(TG_TABLE_NAME AS VARCHAR(64)), CAST(OLD.id AS VARCHAR(128)));
    ELSE
        PERFORM logging(TG_OP, CAST(TG_TABLE_NAME AS VARCHAR(64)), CAST(NEW.id AS VARCHAR(128)));
    END IF;

    RETURN NULL;
END;
$$ LANGUAGE PLPGSQL;
```

```
CREATE OR REPLACE FUNCTION logging(action VARCHAR(64), entity VARCHAR(64), entityId VARCHAR(128))
RETURNS VOID AS $$
BEGIN
        INSERT INTO LOGGING (action, entityId, entity, timestamp)
            VALUES (action, entityId, entity, FLOOR(EXTRACT(epoch from now()) * 1000));
    RETURN;
END;
$$ LANGUAGE PLPGSQL;
```

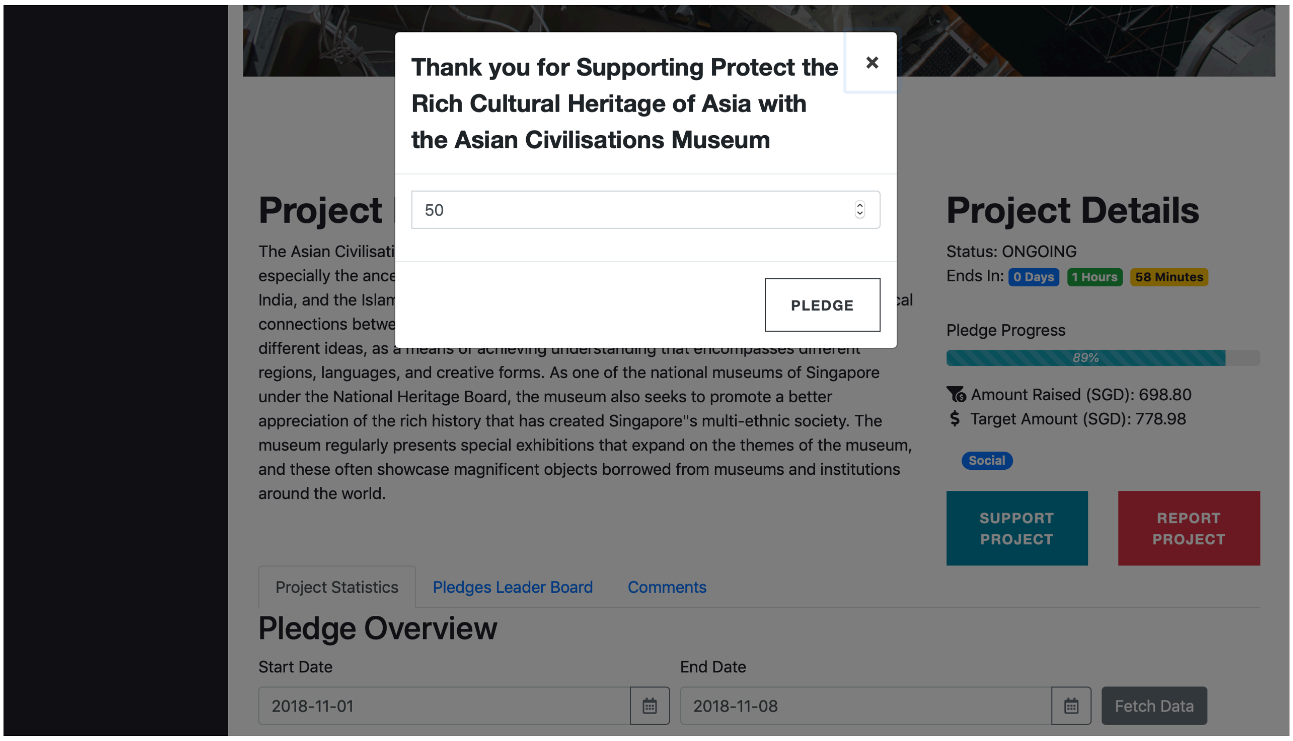
(2) Auto refund project when project is banned (Trigger: refundPledges, function: autoRefundProjects)

```
CREATE TRIGGER refundPledges
AFTER UPDATE
ON PROJECT
FOR EACH ROW
EXECUTE PROCEDURE autoRefundProjects();
```

```
CREATE OR REPLACE FUNCTION autoRefundProjects ()
RETURNS TRIGGER AS $$
DECLARE
BEGIN
IF NEW.isbanned = TRUE THEN
UPDATE PLEDGING_PLEDGES SET isrefunded = TRUE WHERE projectid = NEW.id;
END IF;
RETURN NEW;
END; $$
LANGUAGE PLPGSQL;
```

6. Web interface

Pledge page:


Thank you for Supporting Protect the Rich Cultural Heritage of Asia with the Asian Civilisations Museum

50

Project

The Asian Civilisati... especially the ance... India, and the Islam... connections betwe... different ideas, as a means of achieving understanding that encompasses different regions, languages, and creative forms. As one of the national museums of Singapore under the National Heritage Board, the museum also seeks to promote a better appreciation of the rich history that has created Singapore"s multi-ethnic society. The museum regularly presents special exhibitions that expand on the themes of the museum, and these often showcase magnificent objects borrowed from museums and institutions around the world.

Project Details

Status: ONGOING

Ends In: 0 Days 1 Hours 58 Minutes

Pledge Progress

89%

Amount Raised (SGD): 698.80

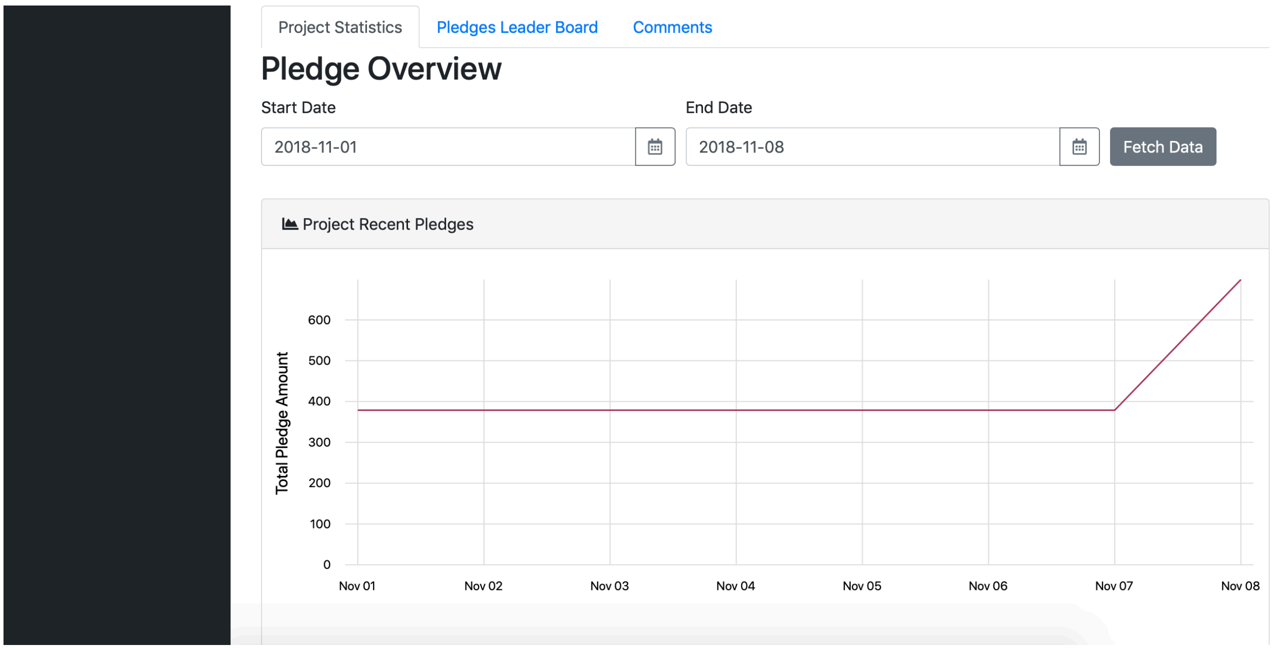
Target Amount (SGD): 778.98

Pledge Overview

Start Date: 2018-11-01

End Date: 2018-11-08

Project pledge overview:



Project Statistics | Pledges Leader Board | Comments

Pledge Overview

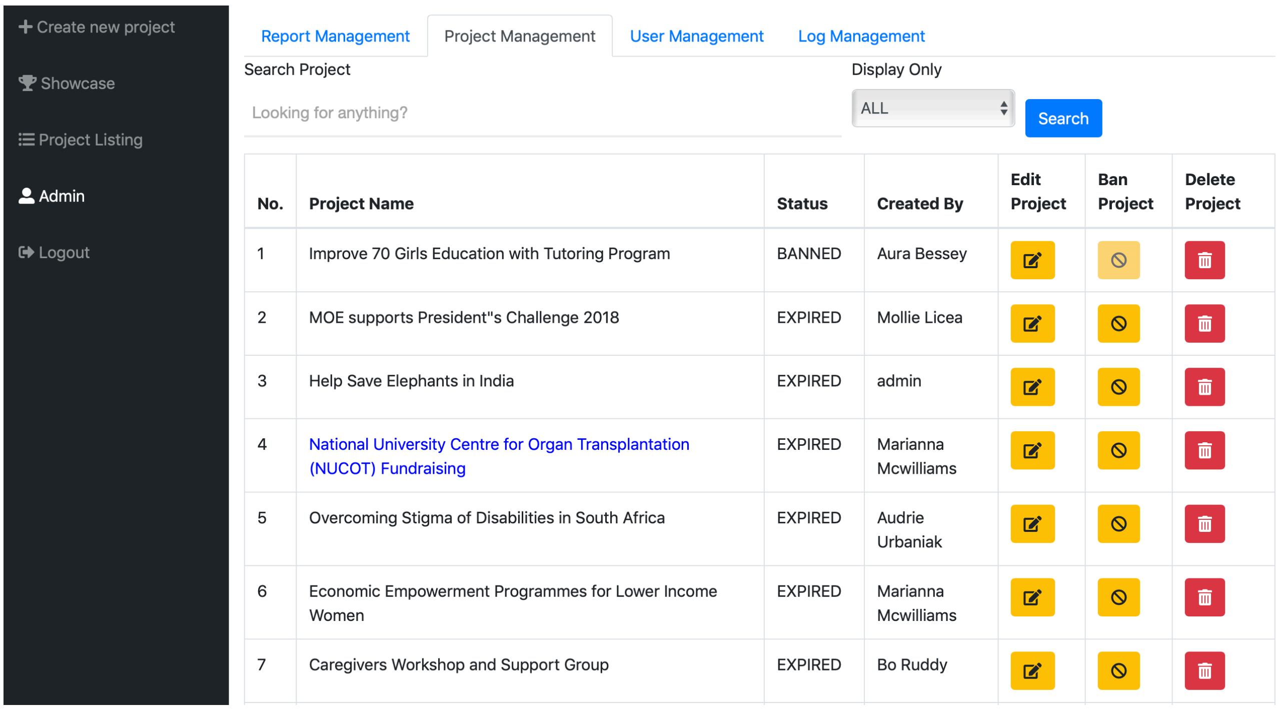
Start Date: 2018-11-01

End Date: 2018-11-08

Fetch Data

Project Recent Pledges

Admin project management page:



Search Project: Looking for anything?

Display Only: ALL

Search

No.	Project Name	Status	Created By	Edit Project	Ban Project	Delete Project
1	Improve 70 Girls Education with Tutoring Program	BANNED	Aura Bessey			
2	MOE supports President"s Challenge 2018	EXPIRED	Mollie Licea			
3	Help Save Elephants in India	EXPIRED	admin			
4	National University Centre for Organ Transplantation (NUCOT) Fundraising	EXPIRED	Marianna Mcwilliams			
5	Overcoming Stigma of Disabilities in South Africa	EXPIRED	Audrie Urbaniak			
6	Economic Empowerment Programmes for Lower Income Women	EXPIRED	Marianna Mcwilliams			
7	Caregivers Workshop and Support Group	EXPIRED	Bo Ruddy			