# CPRG-352 - Web Application Programming

## Lab 6

## Topic: JDBC

This lab is intended to be completed over two weeks.

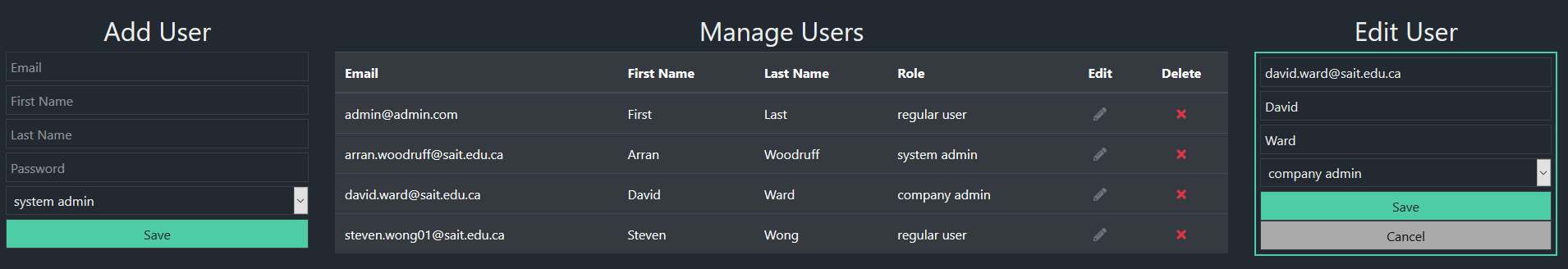
## Description

We are going to be building a 3-tier MVC Web Application from scratch with a MySQL database. The application will be used to manage users. The web application must have the ability to view all users, add a user, update a user, and delete a user. A user must have the following attributes:

* email (primary identifier/key of a user)
* active (boolean to determine if the user account is active)
* first name
* last name
* password
* role (one of system admin, regular user, or company admin)

For updating (editing) a user account, all attributes are editable except for email.

The user interface design is completely up to you. Recommendations: checkbox for active, drop-down list for role, and textboxes for everything else. You do not need to show all of the user attributes for “view all users” functionality; you may show just the first name and last name, if you wish. Below is what last year’s UI looked like for the strongest team. (Please note that there was no active field last year.)



There is no authentication (login) or authorization (permissions).

# Part A: Database

The following SQL script goes in a file called database/userdb.sql within the NetBeans project. Feel free to customize the script as much as you want.

DROP SCHEMA IF EXISTS `userdb`;

CREATE SCHEMA IF NOT EXISTS `userdb` DEFAULT CHARACTER SET latin1;

USE `userdb`;

-- -----------------------------------------------------

-- Table `userdb`.`role`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `userdb`.`role` (

`role\_id` INT(11) NOT NULL,

`role\_name` VARCHAR(25) NOT NULL,

PRIMARY KEY (`role\_id`));

-- -----------------------------------------------------

-- Table `userdb`.`user`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `userdb`.`user` (

`email` VARCHAR(40) NOT NULL,

`active` TINYINT(1) NOT NULL DEFAULT '1',

`first\_name` VARCHAR(20) NOT NULL,

`last\_name` VARCHAR(20) NOT NULL,

`password` VARCHAR(20) NOT NULL,

`role` INT(11) NOT NULL,

PRIMARY KEY (`email`),

CONSTRAINT `fk\_user\_role`

FOREIGN KEY (`role`)

REFERENCES `userdb`.`role` (`role\_id`));

INSERT INTO `role` VALUES (1, 'system admin');

INSERT INTO `role` VALUES (2, 'regular user');

INSERT INTO `role` VALUES (3, 'company admin');

INSERT INTO `user` (`email`,`active`,`first\_name`,`last\_name`,`password`,`role`)

VALUES ('cprg352+admin@gmail.com', true, 'Admin','Admin', 'password', 1);

INSERT INTO `user` (`email`,`active`,`first\_name`,`last\_name`,`password`,`role`)

VALUES ('cprg352+admin2@gmail.com', true, 'Admin2','Admin2', 'password', 3);

INSERT INTO `user` (`email`,`active`,`first\_name`,`last\_name`,`password`,`role`)

VALUES ('cprg352+anne@gmail.com', true, 'Anne','Annerson', 'password', 2);

INSERT INTO `user` (`email`,`active`,`first\_name`,`last\_name`,`password`,`role`)

VALUES ('cprg352+barb@gmail.com', true, 'Barb','Barber', 'password', 2);

# Part B: Models

Implement models.User and models.Role based on the database design.

# Part C: Data Access Layer (object relational mapping)

In the package dataaccess, implement the following classes:

* ConnectionPool.java
* DBUtil.java
* UserDB.java
* RoleDB.java (only getAll() is needed, but feel free to implement all CRUD methods)

Add a resource to the context.xml file for the connection pool.

# Part D: Business Layer

In the package services, implement the following classes:

* UserService
* RoleService (again, only getAll() is needed)

# Part E: Controller

In the package “servlets”, implement UserServlet.

# Part F: View

In the folder /WEB-INF, implement users.jsp.

# Part G: Code review

Review your coding style and ensure all code has the same style.

# Part H: QA

Test the application fully.