# Design Manual

## Introduction:

This design manual outlines the structure, components, and guidelines for creating the University Management System, a web-based application to support admins, professors, and students. The system is built using the Django web framework and MySQL, with the user interface created using Django templates.

## Security Assurance and User Administration:

All users will already have a premade account and will know their credentials. Users will be classified into three groups: admin, instructor and student. Upon starting the application, the landing page will be located at "/univdb" and this is where the login page is. The login page has a username and password text field where the user can then input their known username and password and then press login. When a user logs in their credentials are passed to the view "handle_login". Here the users information is authenticated using the function *authenticate* from *django.contrib.auth* which verifies the login credentials are a valid user. Then based on the user, their group is checked to see whether they are an admin, instructor, or student and based on that they are directed to their respective home page. When the instructor info is passed to their homepage, it is passed in the session so it is safe. On each home page there is a logout button that directs the user back to the login page.

## User Types:

### Admin:

Admins are automatically redirected to "univdb/admin" when logged in where they're able to use F1, F2, and F3. The page uses the template "univdb/admin/form.html" in a view called "index_admin" to place the initial template on screen.

1. **F1.**
   This feature uses the view "instr_ordered" to get the selected radio button that the admin chooses to sort the instructors by. When the submit button is clicked, placed on the screen by the "form.html" template, the admin is redirected to "univdb/admin/instr_ordered" where the view queries the table described in models.py and uses the template "univdb/admin/instr_ordered.html" to place each row of the queried table on screen. Each row contains the instructor id, name, department, and salary.

2. **F2.**

Feature 2 uses the view “salary_stats” to resolve the query. No input from the admin is required, there is only a button placed on screen in the “form.html” template which will redirect the admin when clicked. The redirected page is found at “univdb/admin/salary_stats” and uses the template in “univdb/admin/salary_stats.html” to place each row of the queried table on screen. Each row contains the department, minimum, maximum, and average salary.

3. **F3.**
   Feature 3 uses the already existing tables, teaches and takes, and also uses the new tables we created, researchfunds and published. The researchfunds table displays the professor id for who is working on the research, title of the research, funds they received, agency providing the funds, beginning date and end date of the research, and manager who approved of the funds. The published table displays the professor id for who published the papers, title of the research the papers are about, number of different papers published for such research, and the year, publication venue and publication date for these papers.

   Feature 3 can be run only as an admin by providing a professor name, and optionally providing the year and/or semester for courses via input boxes on the “form.html” page. Then the admin can hit the ‘Submit’ button found in “form.html” where they will be brought to the link “/univdb/admin/professor_performance”, outlined by the html page “professor_performance.html” where data is provided based upon the admin’s previous inputs. The provided data is formatted in a table and includes the number of courses, number of students, total funding, and published papers.

**Professor/Instructor:**
**Professors** are automatically redirected to “univdb/instructor” when logged in where they’re able to use F4 and F5. The page uses the template “univdb/instructor/home.html” in a view called “instructor” to place the initial template on screen.

4. **F4.**
   Feature 4 uses the view “course_prof” to resolve the query. No input from the admin is required, there is one of two “search” buttons placed on screen in the “home.html” template which will redirect the admin when clicked. The redirected page is found at “univdb/instructor/course_prof” and uses the template in “univdb/instructor/course_prof.html” to place each row of the queried table on

screen. Each row contains the course_id, sec_id, semester, year, and number_of students_enrolled.

The professor that is being queried is the currently signed in professor stored under the request.session[‘user’]. The query uses the prof_id given by the instructor object. Then it uses an embedded sql query that uses the teaches and takes tables.

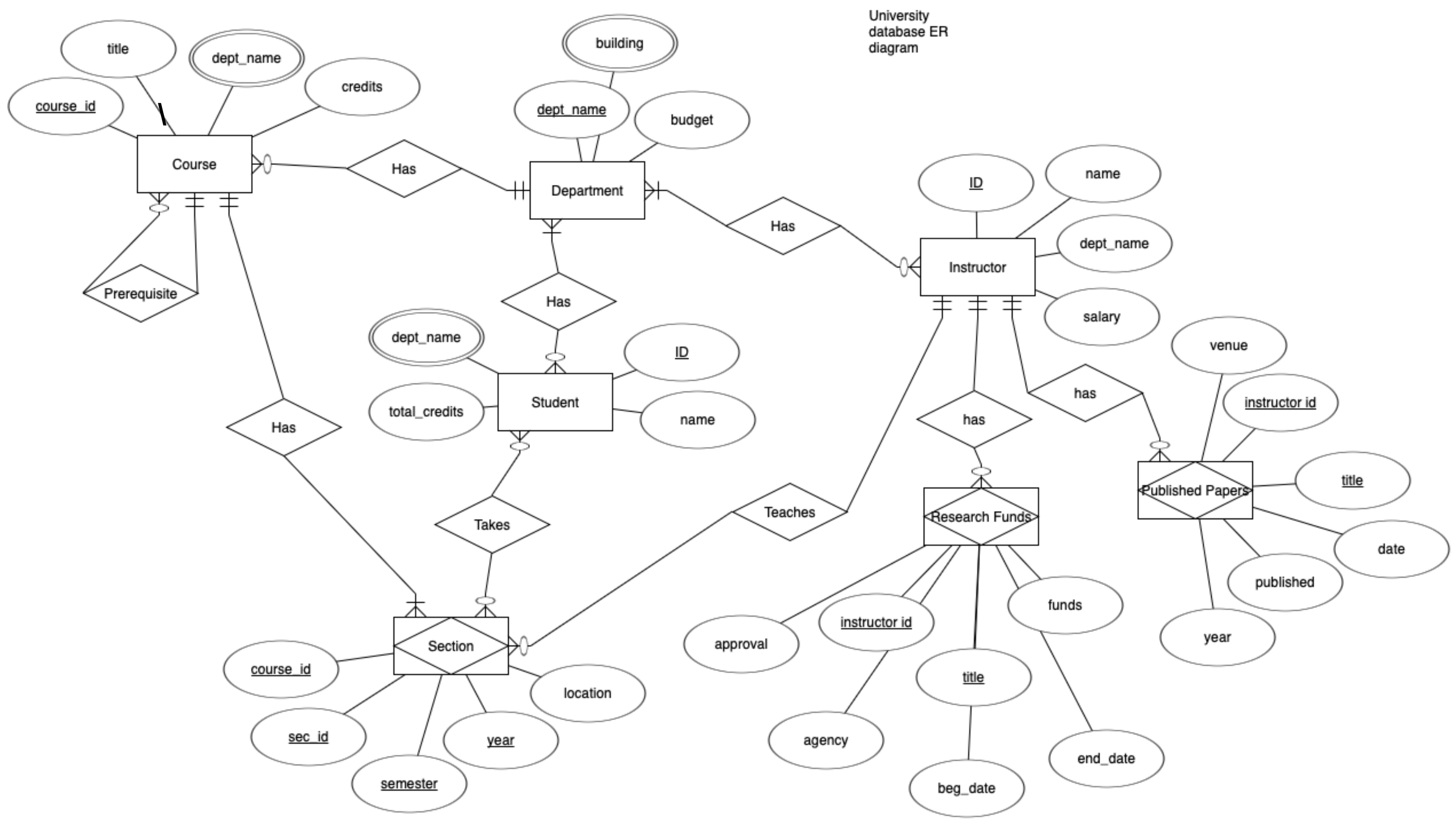
5. **F5.**
   Feature 5 uses the view “student_list” to resolve the query. No input from the admin is required, there is one of two “search” buttons placed on screen in the “home.html” template which will redirect the admin when clicked. The redirected page is found at “univdb/instructor/student_list” and uses the template in “univdb/instructor/student_list.html” to place each row of the queried table on screen. Each row contains the name, semester, and year of each student.

   The professor that is being queried is the currently signed in professor stored under the request.session[‘user’]. The query uses the prof_id given by the instructor object. Then it uses an embedded sql query that uses the teaches, takes, and student tables.

**Student:**
Explain Student.

6. **F6.**
   Feature 6 uses the view “index_student” and “course_offerings”. When a student logs in the land on the page with the view “index_student” located at “univdb/student”. They are then prompted to use two dropdown select menus to choose the desired year and semester for the list of courses that will be created. Upon choosing the desired year and semester they can then press the button generate which navigates them to the page with the view “course_offerings” located at “univdb/student/courses”. Here, using Python’s mysql connector a query is generated based on the input and is then executed to get the result. The result is shown in a table with each row containing dept_name, course_id, title, sec_id, building, room, and credits. The user has the option to press the back button to go back to the student home page and generate a new list of courses.d

## ER Diagram of University database:



## Database schema:

**Department(**dept_name, building, budget)
**Course(**course_id, *dept_name(Department)*, title, credits)
**Prerequisite**(*course_id, prereq_id)*
**Instructor(**ID, *dept_name(Department)*, salary, name)
**Student(**ID, *dept_name(Department)*, credits, name)
**Section(**course_id, sec_id, semester, year, location)
**Takes(***student(Student), section(Section)*)
**Teaches(***instructor(Instructor), section(Section))*
**Publication(**instructor_id, title, venue, date, year, published)
**ResearchFunds(**instructor_id, title, funds, beg_date, end_date, agency, approval)