A design description of the chosen design:

Vi har oprettet en entityklasse for hver klasse i diagrammet. På disse entity klasser har vi anvendt JPA's one to many annotation, join column annotation, og googles expose annotation. JPA har vi anvendt til at mappe mellem databasen og vores backend. Vores backend består af en facade, der anvender en EntityManager, og et en forbindelse til vores REST-API. Vores REST anvender vi til at kommunikerer med vores HttpServer via http protokollen. Webapplicationen anvender AJAX til at lave sine rest-kald under overfalden for at få leveret vores data.

A section explaining your test strategy and test results:

Vi lavede isolationstests på nogle af vores entity klasser og så lavede vi også en integrationstest på entity klassen person for at teste om vi kunne tilføje personer til vores database. Disse tests fik vi 100% .

Så har vi testet alle metoderne i vores facade klasse. Først har vi lavet en isolationstest for at tjekke at metoderne virker isoleret. Så laver vi en integrationstest af facaden, hvor vi tester de forskellige metoder op mod en database. En af metoderne gav nogle problemer, når vi skulle teste den og det eneste der løste det, var at gøre RoleSchool klassen ikke abstract. Udover det så virkede alle vores tests.

A section stating who did what:

Damjan: Jeg lavede den første opgave og var med til at lave entity klasserne. Jeg lavede isolationstests og html. Jeg var med til at lave index(html), script og webservice sammen med Vuk og Jakob. Lavede lidt bootstrap css.

Vuk: Jeg lavede testcases med Jakob - integrationstests. Jeg lavede interface og facade klassen. Jeg har også lavet personfacadetesten. RestAPI lavede jeg sammen med Jakob. I RestAPI lavede jeg en ny handler. AddRoleButton i jscripts. Dette var til for at kunne tilføje en rolle til en teacher/student.

Jakob: Jeg har i samarbejde med Vuk udarbejdet isolationstest af Person klassen, samt integrationstest af Person klassen og databasen. Jeg hjalp til med entity klasserne, og fik rettet dem til, så de rigtige relationships blev oprettet, samt visibility imellem disse. Dernæst lavede jeg facaden og det tilhørende interface. Til denne facade oprettede jeg en mockFacade, som jeg lavede tests på. Både isolation- og integrationstest.

Jeg hjalp til med at få addRole metoden til at fungere. Efter vores backend var lavet, begyndte vi på vores REST-API, som jeg udarbejdede i samarbejde med Vuk og Damjan. Dette inkluderede både rest og webservicen, altså html siderne samt javascript.

Til sidst har jeg ryddet lidt op i koden og gjort det muligt at tilføje roller til personer via webinterfacet.

Youssef: Jeg var med til at lave de entityklasser sammen med Damjan.

A description of what you have added, apart from step 1-7 in this document:

Vi har ikke tilføjet noget.

A description of the strategy used to implement inheritance and why this strategy was chosen:

Grunden til vi har valgt JOINED fremfor SingleTable, er fordi at vi regner med at det er en stor database vi har. Vi skal have tabeller for hele skolen jo, så det hjælper ikke at vi har én tabel hvor en student, lærer, og en lærerassistent er i, da der kommer til at være for mange null-referencer, og det er spild af plads i databasen. Samt at vi har disse 3 tabeller giver et større og nemmere overskud at se på, da det er nemmere at forstå.