Department of Electrical and Computer Engineering
ECSE 321 Introduction to Software Engineering
Winter 2020

Project overview

Your local pet shelter decided to create a website and an app to facilitate the pet adoption process and has asked you to help design the software system for them. Through the website and the app, users can post advertisements about pets they wish to put up for adoption or look for pets they wish to adopt. Everyone using the website or app must register and create a profile before they can use it. The shelter itself can also post advertisements for pets available for adoption on their premises. There is no payment associated with the process, however, users are given an option to donate to the pet shelter if they wish.

In teams of five students, you will gather requirements, design a multi-tier software solution to satisfy those requirements, implement the system, validate that the system is satisfying the requirements, and develop a release pipeline to automate the software delivery process.

Initial user stories

People looking for pets to adopt:

- Have to sign up before they look for pets to adopt.
- Are required to provide relevant information about themselves and their homes.
- Can browse the profiles of available pets to adopt.
- Can apply to adopt a selected pet.
- Can ask for recommendations from the pet shelter.

People putting up their pets for adoption:

- Have to sign up before they can put their pets up for adoption.
- Provide a description of the pet and share photos.
- Can provide a reason for putting up their pet for adoption.
- Can see a list of candidates applying to adopt their pet and select the final adopter.
- Can ask for recommendations from the pet shelter.

The pet shelter:

- Can reply to questions from adopters and owners.
- Can accept donations.

Scope of the project

Your application must support the scenarios described in the user story *for every stakeholder.* All functionality of the system needs to be accessible via the web frontend for respective stakeholders. In addition, a mobile (Android) frontend shall allow the execution of the most important functionality for the given stakeholder, i.e. it shall have both read and write access to the backend via RESTful service calls. External systems or services are not required to be integrated.

Technological constraints

Your project should adhere to the following technological constraints:

1. For each sprint, your team must
   1.1. Provide project backlog using GitHub Projects.
   1.2. Use issues in GitHub to track *development*, *release engineering*, and *documentation* tasks.
   1.3. Define milestones of the project for each deliverable and assign all issues created during a sprint to its corresponding milestone.
   1.4. Provide documentation (e.g. meeting minutes with key decisions, effort table, models, supplementary images) using the wiki pages of the GitHub repository.
2. Starting from Sprint 1 (Database), your team must
   2.1. Use UML Lab to create a domain model.
   2.2. Implement a persistence layer using a Postgres database.
   2.3. Use the ORM technology Hibernate to map objects to database concepts.
   2.4. Create a Spring/Spring Boot project.
   2.5. Configure a build system using Gradle.
   2.6. Use a Continuous Integration process using Travis CI to build and test the database layer.
3. Starting from Sprint 2 (Backend), your team must
   3.1. Implement RESTful web service using Java Spring/Spring Boot.
   3.2. Provide a suite of unit tests for the backend using JUnit.
   3.3. Deploy the project as a Heroku application in addition to the constraints above.
4. Starting from Sprint 3 (Web), your team must
   4.1. Implement the web frontend using Vue.js.
   4.2. Integrate services (functionality) developed by other teams.
5. For Sprint 4 (Android), your team must
   5.1. Implement the mobile frontend using the Android SDK but without the need for continuous integration and deployment for the Android frontend.

A team may choose a technology different from the recommended ones in case of items 2.3, 2.4, 2.5, 3.1, 3.2, and 4.1, but no technical support will be provided. All other technological constraints need to be respected.

Deliverables

You will deliver the system in four main iterations during the term. The corresponding deliverables are to be submitted at checkpoints throughout the term as described below. This section gives an overview of the deliverables including their due dates. More details for every deliverable will be available.

Deliverable 1 – Requirements, Domain Modeling, and Database Design (10%) (due February 9, 11:59pm)

- Requirements, Actors and Use cases
- Domain model
- Database design
- Test cases for persistence layer

Deliverable 2 – Backend and Quality Assurance (10%) (due March 1, at 11:59pm)

- Backend implementation as RESTful services
- Documentation of RESTful services
- Suite of unit tests for backend

Deliverable 3 – Web Frontend and Architecture (10%) (due March 22 at 11:59pm)

- Architecture model
- Implementation of Web frontend

Deliverable 4 – Mobile Frontend and Availability (10%) (due April 5 at 11:59pm)

- Implementation of mobile frontend

Deliverable 5 – Presentation (5%) (March 31, April 2, and April 3, upload due on April 3 at 11:59pm)

- Each group will give a presentation about their project to the class
- Final presentation slots will be assigned by a random draw after specifying available slots
- You may miss at most one presentation slot