National Technical University of Athens, *Software Engineering* Semester Project, 2022-2023

Topic

The thematic field of the work is the management of "smart surveys" for conducting online surveys of all kinds. A survey is considered "smart" when each subsequent question and its answers can be determined by the response to the previous one. Both the questions and the answers, as well as the transition rules, must be parametrically defined. This means that the user should be able to specify the options and paths, and the software should initialize and execute through a suitable interface for data collection. As one can easily observe, there are several web services that do something similar, such as *Google Forms*, *SurveyMonkey*, *Typeform*, *KwikSurveys*, *SurveyPlanet*, and others.

There are two categories of questions: those related to the profile of each participant in the survey and those that constitute the actual content of the survey. The responses to questions in both categories are made available to the system user in a format that allows processing with suitable software (e.g., *Excel*, *PowerBI*, *SPSS*, etc.).

Request Formulation

The subject of the work is the development of a software application for the configuration and execution of "smart surveys," which we will name *intelliQ*. The application will be used to collect data from such surveys in various research projects. The work includes identifying and specifying requirements, architecture and detailed design, implementation of selected functionalities, testing, and, of course, documentation of all these aspects. The requirements specification and design, code management, testing, as well as project administration, will be done using appropriate tools. The entire set of deliverable documents will be generated automatically, meaning there will be no manual writing using tools like *Word*, etc.

The identification of stakeholders and the functions that the system will include will be carried out through discussions in the course. Upon completion of these discussions, a list of stakeholders and functions will be available in the course area on *helios.ntua.gr*. From this list, some functions will be implemented by all teams, while others can be selected by each team, according to instructions that will be announced.

Sections of the application that you will construct are as follows:

1. A backend subsystem, which will support functions for managing the structure of surveys, collecting, entering, and exporting responses. These functions will be available through a REST API, the specifications of which will be provided to you.

2. A Command Line Interface (CLI) application for the above functions, which will operate as a client of the REST API provided by the back-end subsystem, allowing the user to perform functions through a shell.

3. A frontend application (preferably web-based), which will provide capabilities for selecting and answering surveys. This application should also act as a client for the REST API.

Teams

The work will be carried out in teams of 4-5 people, each of which will implement the complete development cycle of the system (requirements analysis, specification writing, design and architecture, implementation and acceptance testing, installation, and operation). Teams with fewer than 5 individuals or up to 6 individuals are allowed, with appropriate adjustments to the deliverables, as explained below.

For collaborative version management of documentation and source code, the use of *GitHub* is mandatory, along with other tools as described below. The formation of teams should be completed no later than 23.10.2022.

Minimum Common Technical Specifications

The system you will develop should support the following features (minimum common specifications):

1. The functions to be implemented will be selected from a catalog that will be announced on Helios, according to the number of members in each team and other instructions provided.

2. The backend will provide a REST API, optionally compatible with the *OpenAPI 3.0* standard. Specifications for the endpoints you will develop will be announced during the implementation phase of the work and will be common for all teams.

3. The language of the user interfaces in the CLI application will be English. The language of the user interfaces will be either Greek or English, with an optional choice.

4. Each team will create test scenarios and will incorporate the execution of the corresponding tests automatically for the functions of the back-end subsystem.

5. It is desirable, in the case of a frontend in a web environment, to support the HTTPS protocol for all interfaces using a self-signed certificate.

These requirements are detailed according to the number of team members, as shown in the deliverables table in the next section.

Technical Requirements – Tools

- ***GitHub*** as the code management environment. Please do not create any repository before the relevant instructions are announced.

- ***Visual Paradigm*** as the UML diagram production tool. You can use the trial version of *Visual Paradigm Enterprise* (free for one month) or the *Community Edition*.
- ***GitHub*** for project management, using the relevant functionality offered there.

Development stack that can be used:

- **Code implementation**: Python, JavaScript with *Node.js*/*Express*. Other options (Java, .NET) are allowed with your declaration.
- **Data management**: one of *MySQL*, *MariaDB*, *PostgreSQL*, *MongoDB*, *Elastic Search*.
- **Frontend**: the choice of frameworks and technologies for the frontend (e.g., JavaScript libraries for charts/graphs on the web) is free.

Deliverables

Deliverable	1-2 team members	3-4 team members	5-6 team members
Documentation			
Stakeholder Req. Specification (StRS) Document	For 1 stakeholder	For 2 stakeholders	For 3 stakeholders
Software Req. Specification (SRS) Document	1 use case	2 use cases	3 use cases
ER Diagrams or NoSQL JSON Schemas	Yes		
UML Activity/State Diagrams	1 use case	2 use cases	3 use cases
UML Class/API Diagrams	Yes		
UML Sequence Diagrams	Yes		
UML Deployment Diagrams	Yes		
UML Component Diagrams	Yes		
Implementation			
Backend Functions	Yes		

Database Dump (SQL or JSON format)	Yes		
RESTful API	Yes		
API Documentation (demo)	Yes		
Command Line Interface (CLI)	No	Yes	
User Interface (Frontend)	For 1 use case	For 2 use cases	
Executable Format	Yes		
Testing			
API Functional Tests	Yes		
CLI Functional Tests	No	Yes	
CLI Unit Tests	No		yes

Deliverable Formats

Deliverable	Format	Filename	Comment
Documentation			
Stakeholder Req. Specification (StRS)	docx or pdf automatically generated from *Visual Paradigm*	strs-softeng-XX.zip	Documents should also be included in the vpp file.
Software Req. Specification (SRS)		srs-softeng-XX.zip	
ER Diagrams or NoSQL JSON schemas	*Visual Paradigm* (vpp) file	softeng22-XX.vpp	One unified vpp file for all diagrams; diagrams from design software or more than one vpp file are not accepted.
UML Activity/State Diagrams			
UML Class/API Diagrams			
UML Sequence Diagrams			
UML Deployment Diagrams			
UML Component Diagrams			

Implementation			
Insertion/Management/Access to Data (Backend)	Source code	(according to the implementation)	Include a README.md file providing a brief description of the application components and the steps required to set up the development environment (libraries, etc.).
Database Dump (SQL or JSON)	SQL or JSON		
RESTful API	Source code		
API Documentation (demo)	*Postman* scripts		
Command Line Interface (CLI)	Source code		
User Interface (Frontend)			
Executable Format (build & deploy your code from source)			
Testing			
Backend Functional Tests	Test scripts	(according to the implementation)	
CLI Unit Tests			
CLI Functional Tests			
API Functional Tests			

Note: XX represents the team number.

Deadlines

Delivery 1: This includes the version of the complete documentation up to that point, as specified in the corresponding table, possibly along with the part of the implementation completed by then. A single vpp file will be submitted. Also, a snapshot of the repository on *GitHub* will be recorded. The submission must be made on the *Helios* system no later than midnight on December 4, 2022.

Delivery 2: This includes the entire work. The deadline is until midnight on the day before the examination.

Grading

The course grade is determined by 50% from the assignment and 50% from the final exam. Besides the submission, the assignment is presented orally by the team members on dates announced after the end

of the examination period. To pass the course, a passing grade is required for both the assignment and the final exam.

The grading of the assignment is based on the following weights:

- Documentation: 35%
- Implementation: 35%
- Testing: 15%
- Tool Usage: 15%
- Overall Impression: 10%

Note 1: Normalization, if required, is done at the task level and not for the entire assignment or exam. The grading of the assignment for team members may vary depending on the data recorded on Helios and GitHub.

Note 2: The final formula for grading is as follows:

- `FinalGrade = ceiling(sumproduct(PartialGrades;weights);1)`

Where:

- `ceiling(7.1;1) = ceiling(7.9;1) = 8`

Any additions, modifications, etc., to the instructions will be announced on *Helios*.