Team Pontelli

**Soft Skills Training System**

**Design Specification Document**

Nathan Anderson

Jonathan Delira

Flor Lopez

JuSeung Park

CS 448

**1 Introduction**

The implementation of a soft skills training application covers most aspects of modern information technology systems. It requires attention to user interface and user experience for both employers and employees. It is a distributed content system and user account management system. It dwells in a publicly exposed web server, with a back end API for mobile devices, and a data management system. This complete data solution also is faced with all the challenges inherent to every similar system.

**1.1 Design Drivers**

The first thing, which drove us to pick our design for the Soft Skills Training System was the request from the Bridge of Southern New Mexico that they want the mobile application on Android platform. Having that restriction made us to pick the platforms for other components of the system. Choosing either having soft skills modules’ data in the application or in the separate database was another issue because the server would not have needed, if we had the data within the application. The decision our team made to have a separate database had a big impact on how we designed our system. Besides, the application, web site, and database will be communicating through the server, so the server has to be reliable over everything else.

**1.2 Acronyms, Abbreviations, Definitions**

Android - A smartphone operating system

Spring MVC - A server framework

Bootstrap - A web design framework

Soft skills - Skills such as social graces, communication, language, personal habits, etc…

APK file - A file that contains application, which can be installed in Android devices

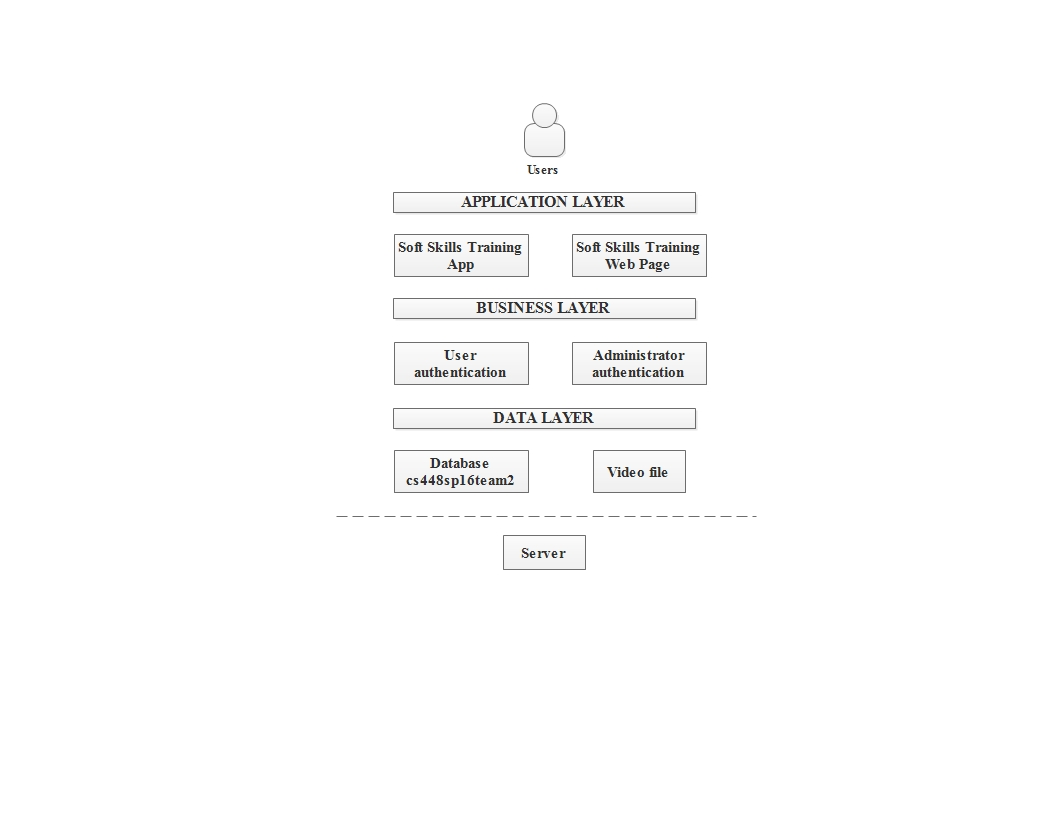
**2 Platform, Languages, and Environment**

The mobile application of the Soft Skills Training System will be implemented with an Android platform because the Bridge of Southern New Mexico asked us specifically to build the app for Android platform. Furthermore, the Android platform is known as the most used smartphone operating system.

Android applications are written in Java, so the application will be in Java. Therefore, the other components will be written in Java except for the web site, so the whole system can be communicating with each component smoothly. The web site will be in HTML and Javascript.

The Bootstrap framework for web development will be used in our web site design, and the Spring MVC server framework will be used for the soft skills training system server. Our team will be using the Android Studio for the app development, and the other components will be developed using Netbeans and Eclipse.

**3 System Architecture**



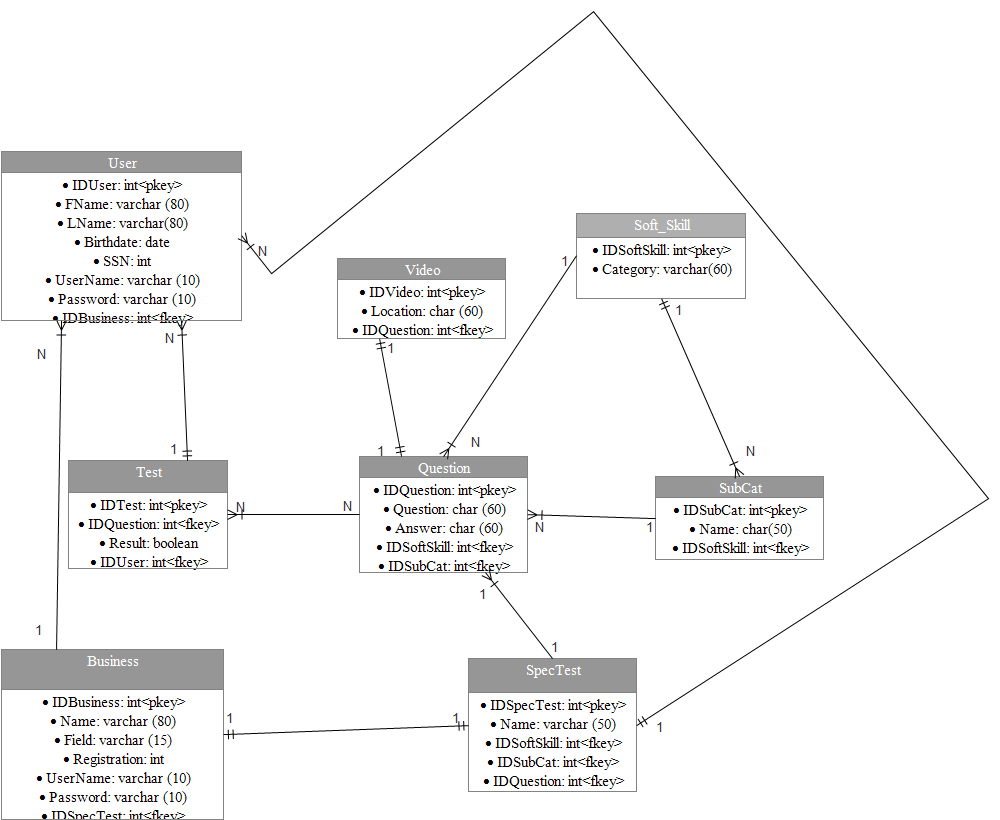
The Soft Skills Training System (application and web page) is based on layered architecture. The data layer will have access directly to the server’s raw data. Inside the data layer there are the database tables and the files containing the videos and images for the training modules. The business layer will be responsible for the authentication. The business layer will be able to modify or change the data indirectly, so the application or the web site will be giving business side for options to do so. Finally, the application layer will provide the user interface for the android application and the web page.

**4 System Design**

domain.png

The domain model represents the system design well. We have the web server hosting a website on the front end, an API for android app backend access, a Database for storage of users and module data, and two separate raw data storage areas for videos and pictures. All the components access the data through the server, and the server will secure, fetch, modify, or remove data from the database. The Spring MVC server framework will secure the connections automatically, and the server will be responsible for checking user informations when users log in. As mentioned above, the data for training modules will be stored in the database, and the application will get the right picture or video from the database. Users will be able to sign up using the application as well.

**4.1 Data Design**



The data that the system is going to handle will be divided into tables according its category. The users will be separated into users (User) and business/employers (Business) each of them will handle personal information to allow identification. The test provided will be general (Test) and specialized (SpecTest); it is important to mention that the specialized test will only exist if a business or employer creates it. The soft skills will be handled by categories (Soft\_Skill) and subcategories (SubCat). Each one of these categories/subcategories will contain questions (Question) and the associated video (Video) will be linked to it according to the number identification of the question. Notice that for purposes of handling the videos it is going to be created a directory containing all of them and in the table is just to be specified the path (location) of each video.

**4.2 Architecture Component Interfaces**

Application flow (3).png

Here is the basic flow of the system. Users primarily interface with the android app. Business clients and system administrators interface with the server. In this diagram the website front end, server back end, and the MySQL database are all housed on the server. The focus of this diagram is to show the entry points of the different users.

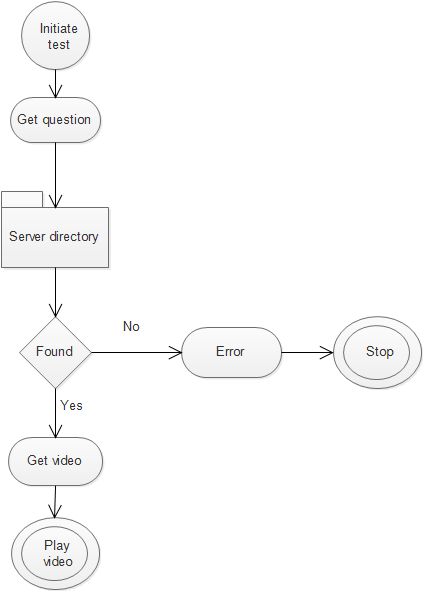
**4.3 Architecture Component Designs**



The web server is built with the spring MVC framework. It is a framework composed completely of plain java objects. It abstracts away the details of the system design and implements best practice structures internally. It has build in security features that protect against common server risks.

The first part of the web server is the front controller. It directs the incoming HTTP Request to a mapping handler. This searches the specified server directories to find a controller that is responsible for the requested URL mapping. Ultimately the controller is the gatekeeper that prevents a user from stepping outside the bounds of a predetermined sequence of pages. It will discriminate user’s access based on authentication and authorization, as well as assemble the required information needed for the respons. In this step the business logic is invoked and the model. Finally, a needed view is specified. The view resolver searches the appropriate directories to find the specified markup files to set up a display for the next user experience. The view resolver is used in the final HTTP response for situations where a web site is to be displayed. If the controller is handling a request for the back end API, then a framework specific call is used send back the needed data directly.

**4.4 Important Interaction Sequences**



For handling videos we are going to use a server directory with the purpose of making easier and faster the management of the database as well as to improve the size of it. Basically, when a question calls its video the system will look in the Video table for the location of the video and open it.

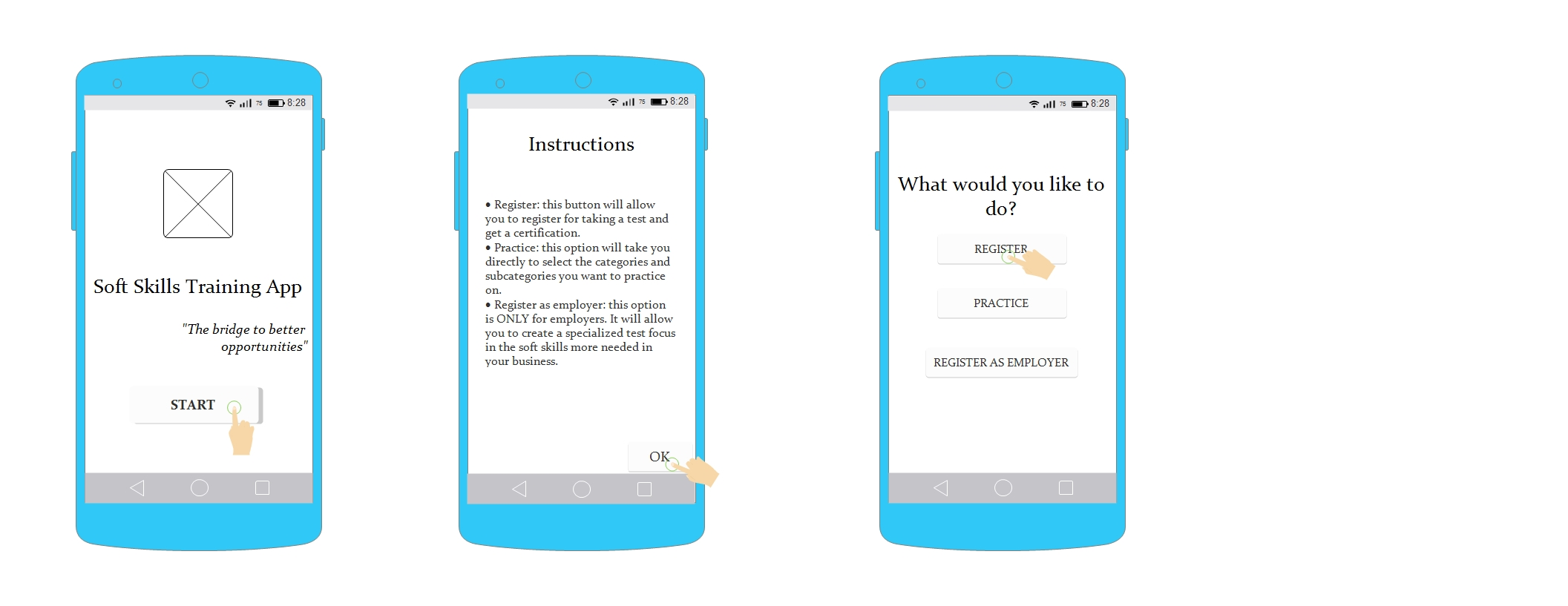
**4.5 Important Object State Diagrams**

Android state diagram.png

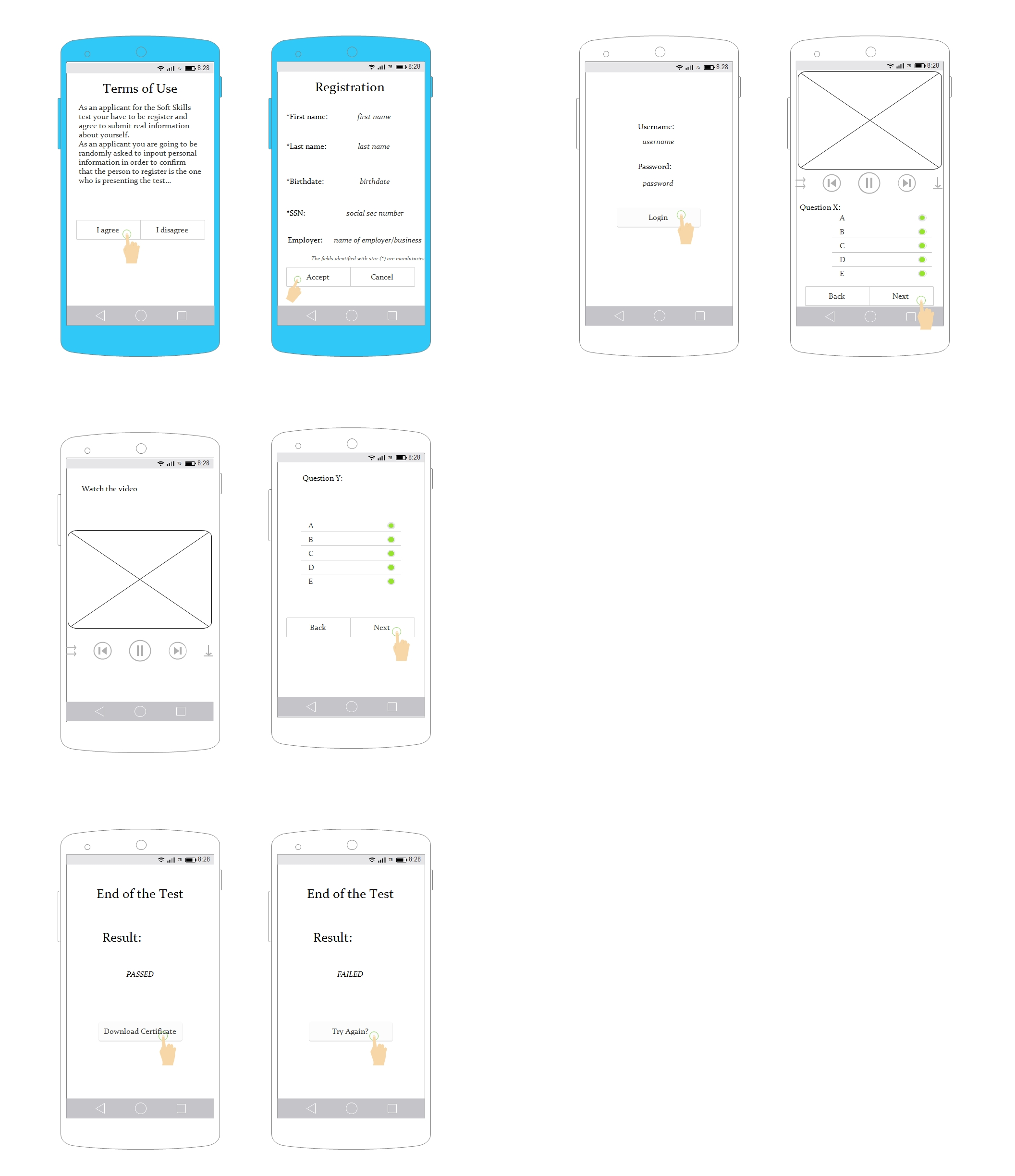
The first time the android application is run the user will see an instructional shadow screen overlaid. In the next menu the user will have their authorization checked. If no special authorization is granted the user will have the option to practice the master list of soft skills. If they had been pre-authorized a test then the test button would appear.

**5 User Interface Design**

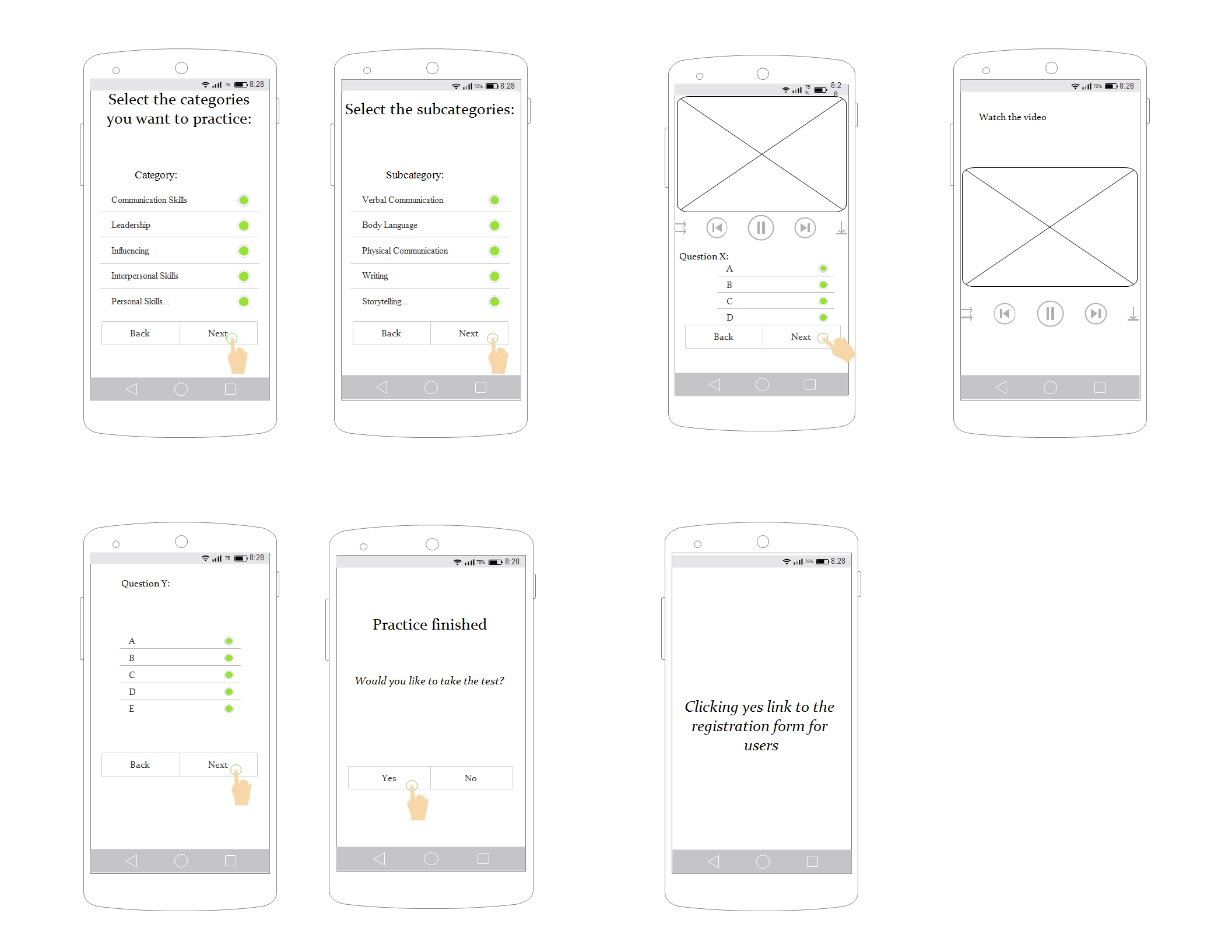
When the user download or uses for first time the application:



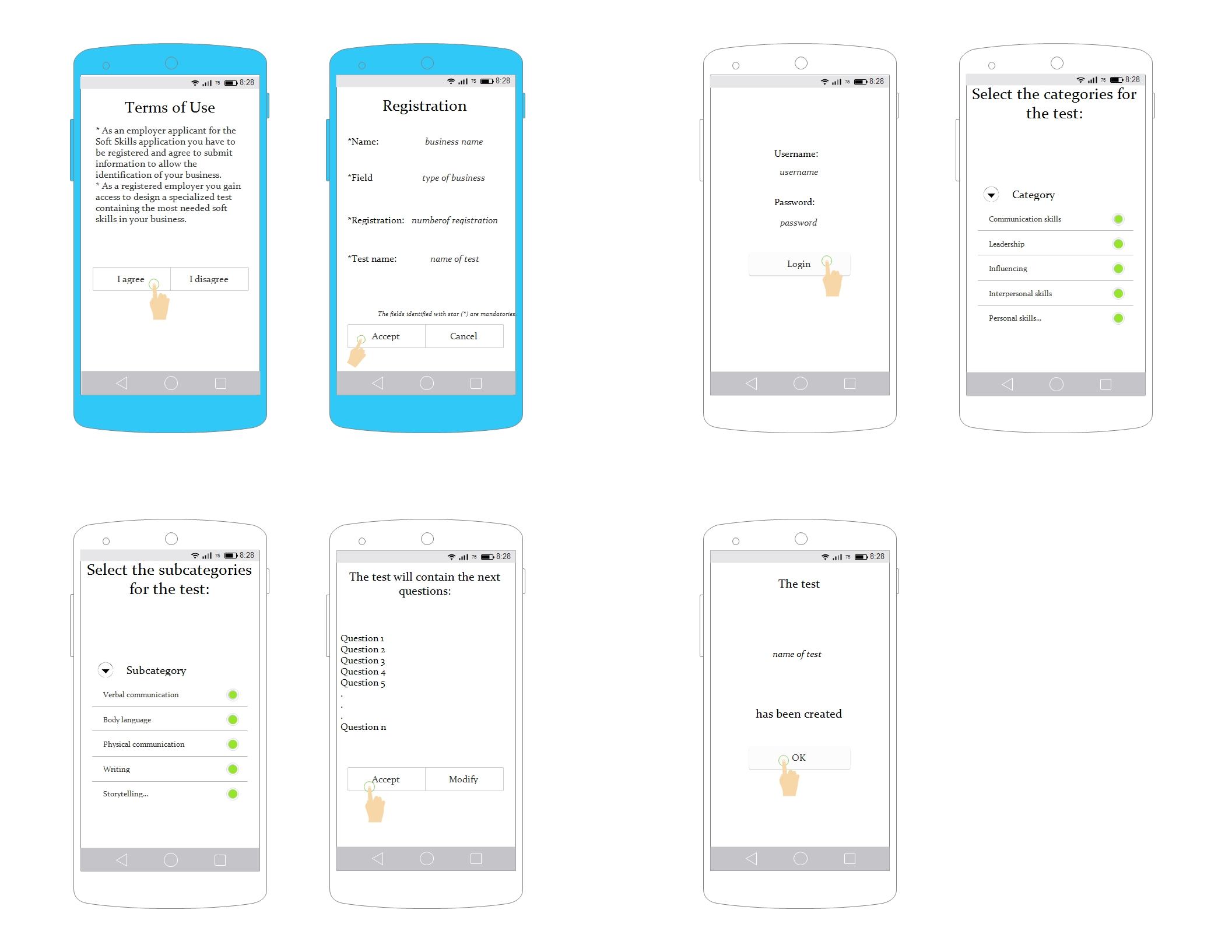
By clicking register the application will have access to the next process where the first two screens will appear only when the user register:

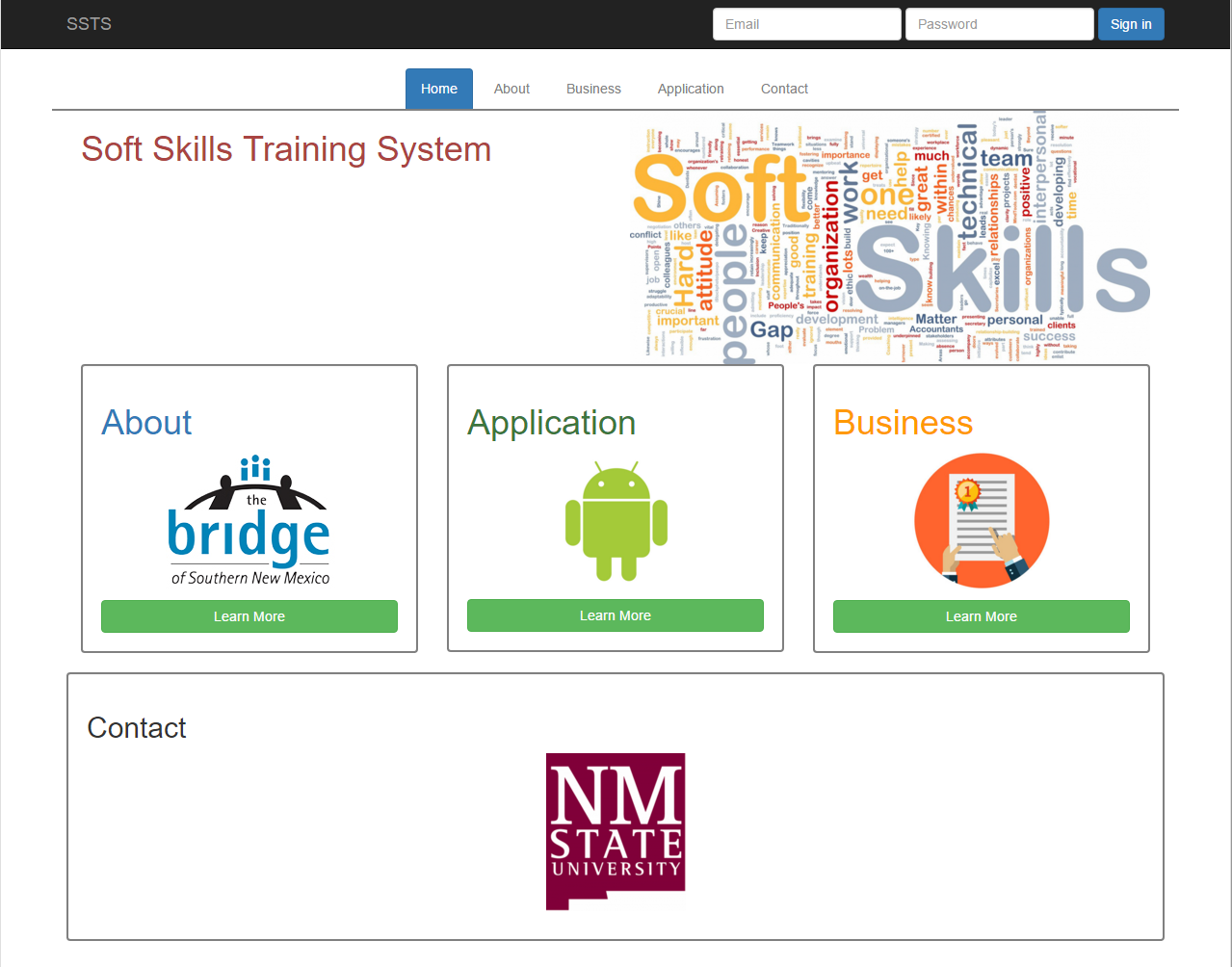


If the user clicks practice the application will send him to the next process:

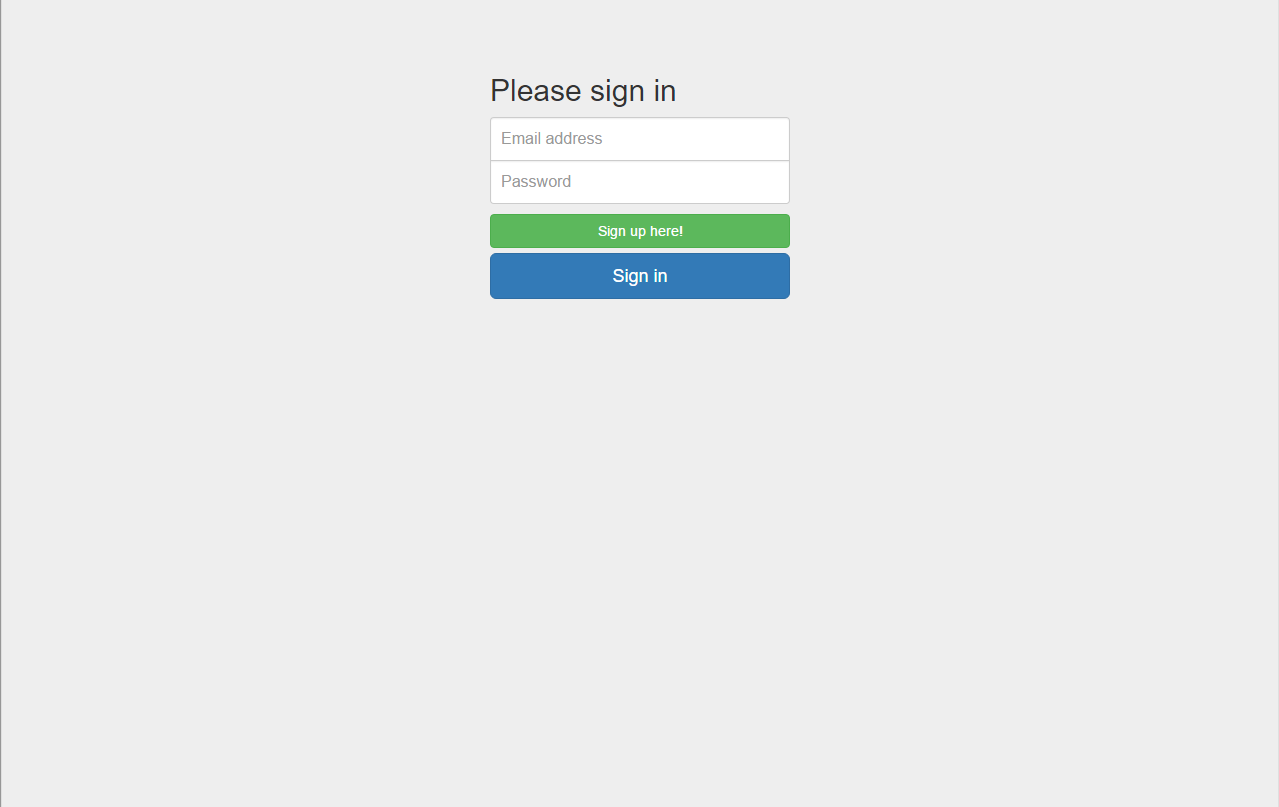


Finally, if the user is an employer/business he has to click the register as employer button and the application will grant access to the next process, the first two screen are appearing just for the registration of the employer /business:





The picture above depicts the front page of the web site. The web page has five tabs, including home, about, business, application, and contact. Each tab will have information regarding the Soft Skills Training System, and the business tab will need an account to access. The business tab will potentially have functionality, such as making sets of tests and viewing the various categories of training modules.



If the user tries to access the business tab, it will lead the user to the login page where the user can sign up or sign in.

**6 Deployment Design**

The server and MySQL database will be in NMSU computer science department, and they do not need to be delivered physically. Our client will get administrator access to the database and web site. The application APK file will be in the database, and the users will be able to download it through the web site. The web site will have a direct download link in the application tab. Once the server is up, the users and the client will be able to access all the system with their own personal computers.

**7 Coding Standards**

We are implementing the Java coding conventions. With comment blocks that will be usable with JavaDocs, classes capitalized camelcase, and functions and instances of classes lowercase.

* **Beginning comments**: will start with a forward slash and asterisk and will end with an asterisk and a forward slash. This type of comments will contain the class’ name, version, date, and copyright notice.
* **Documentation comments:** this comments are delimited in the same way that the beginning comments (/\*\*...\*/) containing describe Java classes, interfaces, constructors, methods, and fields and they should go before the declaration.
* **Class/interface implementation document:** it is delimited by /\*..\*/ and encloses general information about the class or interface.
* **End-Of- Line comments:** these comments are delimited by two forward slashes (//) and only comment a line or a part of it.
* **Indentation:** Lines should not be larger than 80 characters. If a line is too long then it has to be break it but only after a comma, or before an operator and indent them by 8 spaces.
* **Variables:** should be declared in the next order: public, protected, and private. Variable names should be undercase and if they are composed by two names the second should start with uppercase without a white space between them.
* **Variables declaration:** It is better to declare just one variable per line at the beginning of the blocks. Also if the variable must be at the beginning of the blocks. Also, if the variable must be initialized it better to do it at the moment of the declaration.
* **Class/interface declaration:** no leaving space between name and parentheses “(“, the open brace “{“ should go right after the closing parentheses in the line of the declaration statement while the closing brace “}” goes in a line by itself.
* **Statements:** Place only one statement per line and close it by a semicolon “;”. If it is a serie of statements enclosed them by braces ( *{ statements }* ).
* **White spaces:** they improve readability. Add them between sections of source files, class/interface definitions, methods definitions, variable declarations and the first statement. Also place blank spaces between a keyword followed by parentheses, and in binary operators.