# 1. Introduction

## 1.1 Purpose

This document will outline the software architecture and design for Human Resource Job Posting and Management System (HRJPMS). Purpose of this document is to provide overviews of the system's design in order to make it easier to understand the system. It intends to show the significant high lever architectural design of the system.

## 1.2 Scope

This document explains the architecture and design of HRJPMS. It will provide a view of how functional requirement specified in Requirement Document will be designed and accomplished. It will also show to the intended audience how different components interact with each other and how application interface will look like.

## 1.3 Intended Audience

Document is written on a technical level and contains terms that address the technical department. Its main audience are developers, testers and project managers.

## 1.4 References

OBSS - 2017 Java Yaz Stajı Yönergesi document

## 1.5 Definitions, Acronyms and Abbreviation

- HRJPMS - Human Resource Job Posting and Management System
- DBMS –Database Management System. A programmable interface which provides a common layer of abstraction between a physical database and a user or external program.
- JPA - The Java Persistence API is a Java specification for accessing, persisting, and managing data between Java objects / classes and a relational database.

- STMP, IMAP,POP3 – Different mail request protocols.
- SQL –
- JDBC
- API
- MVC
- LDAP
- ER
- DFD

## 2.Requirements

The overall requirements are extracted form provided document at the begin of internship. In summary from that document, the main requirements are listed below.

There will be two types of users in the system: administrators and normal users. Authorization and authentication module shall be implemented to handle different user types. Administrator will have to log in using an LDAP server while normal users will be authenticated using LinkedIn API.

Administrator shall create a new job advert, make it active, disable it or select a time and date to perform the latest. Job adverts must be viewed by everyone accessing the page, but whenever the candidate decides to apply to a specific job, he will need to register using his LinkedIn profile details. All administrators, part of the system, can see a list of all users that have applied to a particular job advert.

Furthermore, they can see candidates profile and all job adverts that the candidate has applied. Also they can process each of the applications made by changing the status of application to either: in process, accepted or rejected. Whenever status is changed, candidate should be notified by email.

Administrators can add users to blacklist. Users added to blacklist can't apply anymore to a job advert and all older application's status are changed to rejected. Every job advert will have a set of skills required and each user will have its own set of skills. Whenever

administrator decides to list all candidates for a specific job advert, they should be sorted from the most matching candidate to the least.

## 2.1 Proposed Solution and Used Technologies

To allow for clean interoperability between different platforms, Java is being utilized as the implementation medium. All functionalities in application logic layer will be written in Java. To make things easier and speed up the development process, Spring Boot will be used since it comes pre-configured and allows to focus more in development phase rather than spending time configuring the environment.

Hibernate will be used in Data Access Layer to manage persistent data. It provides support for Object - Relational Mapping for the SQL based Databases, which is the main advantage over JDBC. Java is OOP based language means it deals with Objects all the time, on the other hand Relational Databases are all about writing data in tables, in JDBC we save data in each column of table separately, where as in Hibernate we can store the object directly in database where each row of the table represents an object. Hibernate comes with 2 levels of cache, so performance improvements over JDBC. Also, hibernate is checked exception free means it will not throw any error if any checked exceptions are not handled, it will handle the checked exceptions automatically which is not the case with JDBC.

In Addition, Java Persistent API and JPA data repositories will be used to avoid the need to write Hibernate Language Query. Data source and main DBMS will be MySQL. It is powerful, free open-source database management system that has been around for years. It is very stable and has a big community that helps maintain, debug and upgrade it. MySQL has one major advantage, since it is free, it is usually available on shared hosting packages and can be easily set up in a Linux, Unix or Windows environment. It is very well suited for small to medium web applications.

Thymeleaf is chosen to be the main view resolver. Thymealef looks more HTML-ish than JSP or other view resolvers, there are no strange tags like in JSP. It drastically reduces the number of steps to design a view. Furthermore, it has a big community and a nice documentation with lots of examples as well as a very stable integration with Spring MVC.

# 3 Architecture

## 3.1 Physical Deployment Diagram

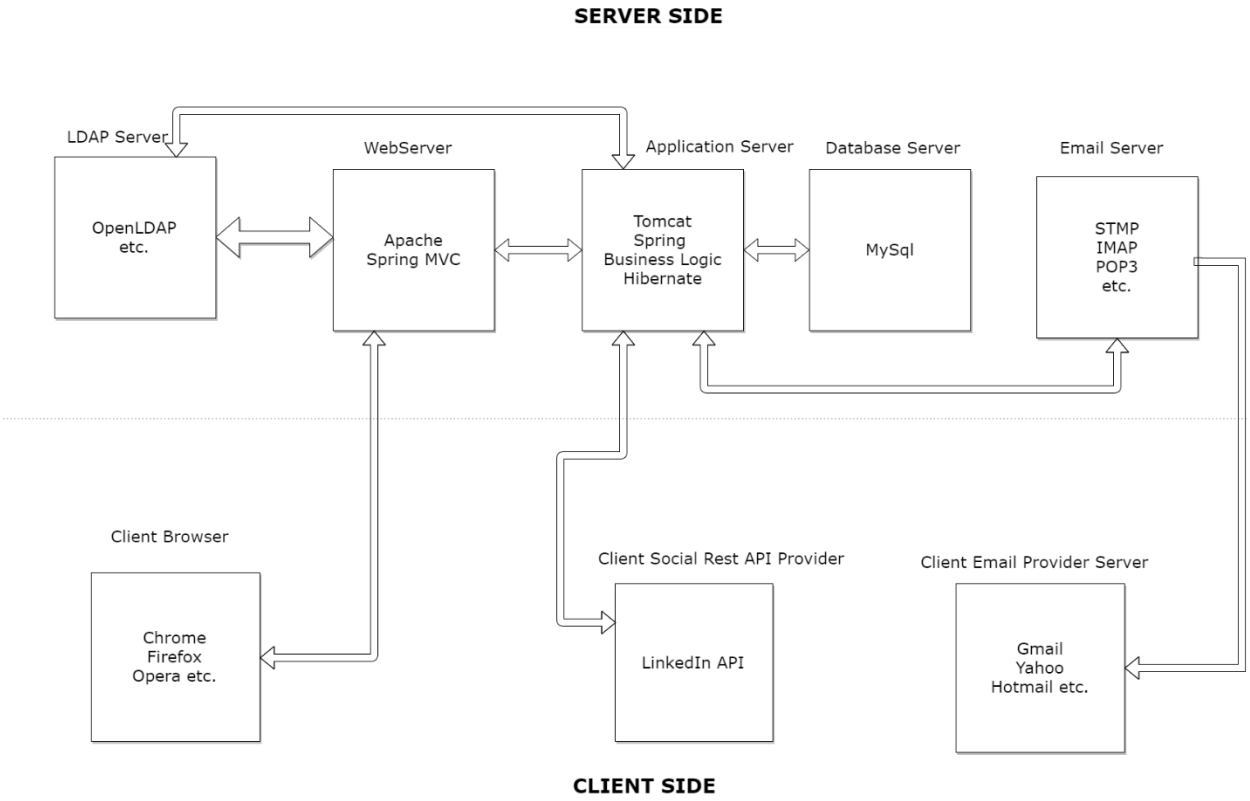



Figure 1: Design Diagram

### Server

- **Web Server**
    - The web server for HRJPMS provides an access and to the application. Its main function is to listen to client requests, process the request, exchange data with Application Server and display the proper view page. All requests to Web Server are made using https protocol on port 8080. Apache is chosen as Web Server since It is usually used in conjunction with Tomcat Application server.

- **Application Server**
    - Application Server is where application logic resides. It is the core of the system. It makes use of Tomcat Servlet server to execute Java written server side code (Servlets).

- **Database Server**

    - Database server is where the Database Engine runs, in our case MySQL InnoEngine. It doesn't have to be a separate physical server but in 3-tier architecture it is mostly a separate machine or a group of machines.

- **LDAP Server**

    - Purpose of LDAP server is to keep in one place all the information of a user (contact details, login, password, permissions), so that it is easier to maintain by network administrators. In HRJPMS system, it is the source of authentication and authorization for HR expert users.

- **Email Server**

    - Email server is a computer system that sends and receives email. It makes uses of different protocols such as STMP, IMAP, POP3 etc. to either send or receive mail requests. In HRJPMS, it is used to send status changes to candidates that have applied to a specific job advert.

### Client and External systems

- **Web Browser**

    - In order to make use of the system, user should use a web browser that is able to process HTML pages.

- **LinkedIn Rest API**

    - The LinkedIn REST API is the heart of all programatic interactions with LinkedIn. In order for HRJPMS system to access LinkedIn member data and/or act on their behalf, they must be authenticated.

- **Email Service Provider**

    - It is required that each user of HRJPMS must have an email address from any provider.

## 3.2 System Architecture

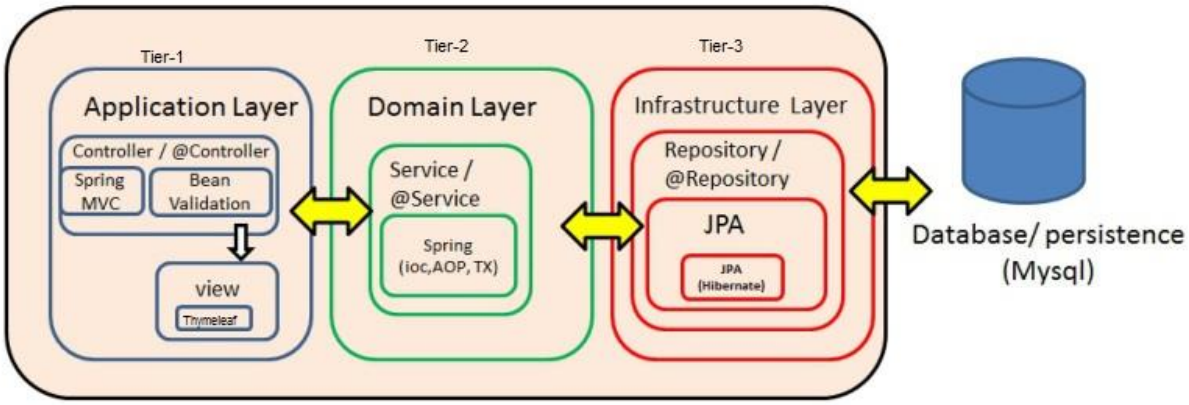
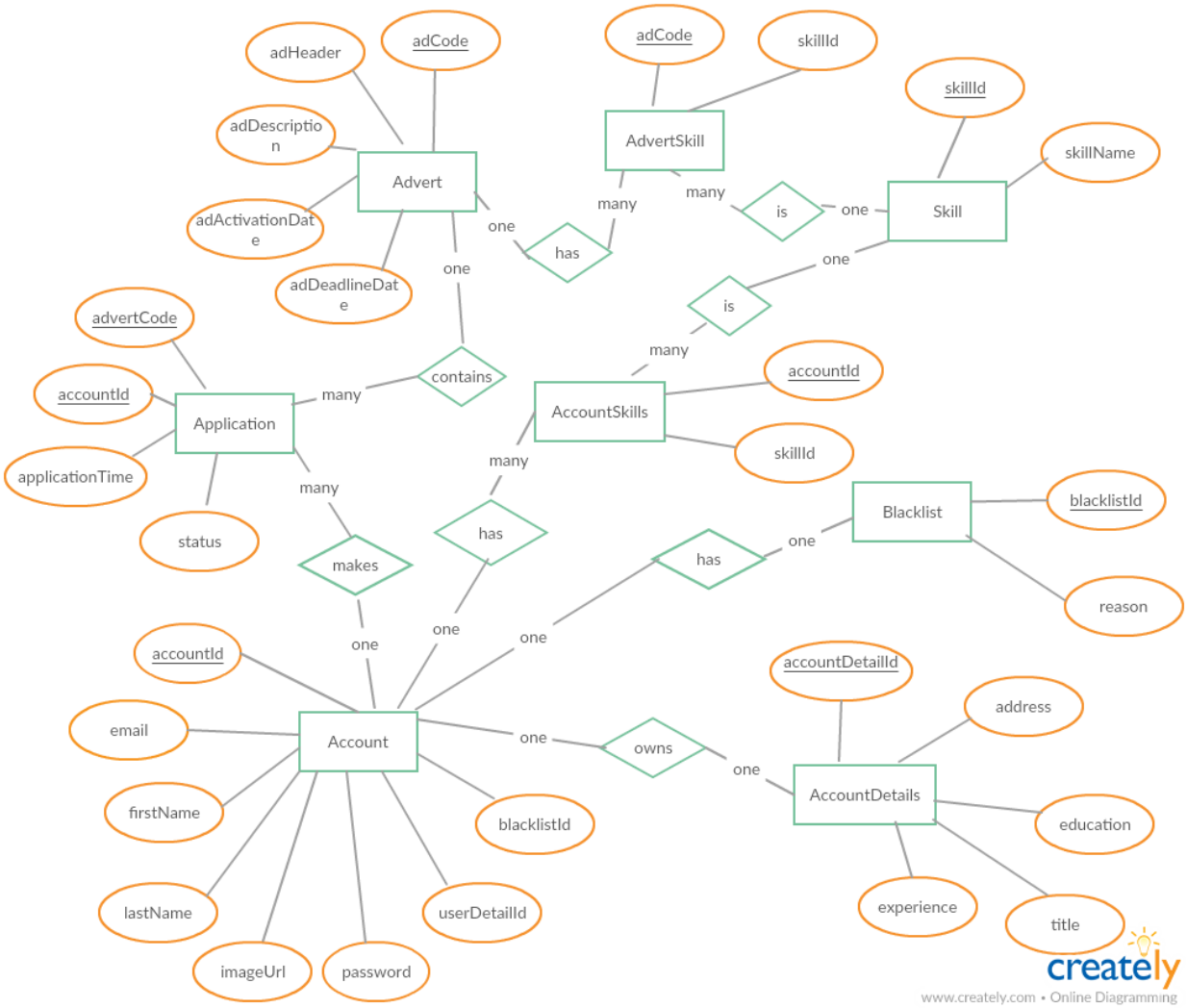



Figure 2: Architecture Diagram

Various implementation with different architectural styles can be found in a standard web application. My first thought was to go with classic Client-Server Architecture where there is only one machine which servers all clients. This type of architecture was immediately dropped because there exists a high risk of security in such types of systems. I decided to go with an architectural model which reduces coupling and is easier to maintain. Three-tier architectural style is an enhanced client-server architecture where interface, process logic and data storage are developed independent from each other. It consists of three main layers: Presentation Layer (User Interface), Application Logic Layer and Data Storage Layer (database). This architecture is based on a linear path of events. All client request has firstly to pass through application logic layer which then processes the data and then submit it to the database. Data flow is bidirectional which means we can either read data or write data from presentation layer to data storage layer through application logic layer. Client interfaces and clients themselves never access data directly from database which makes this architecture style very secure. Even more any changes made to any of the layers don't affect other layer as long as each layer input and output methods doesn't change. This increase scalability and reusability.

Furthermore, this architectural model allows to place both logic layer and data layer in a same machine but it's highly recommended to keep each layer in separate machine's (servers) for security purposes.

## 3.3 ER Diagram




`

Figure 3: ER Diagram of the system

The following is an explanation of each entity in the ER diagram (figure 3).

- Account, represents the user (candidate) which will apply to a certain job advert, has the following attributes
    - accountId – unique ID generated by DBMS engine
    - email – unique attribute
    - firstName
    - lastName
    - imageUrl - path to LinkedIn user image avatar
    - password – generated SHA1 password for registered user
    - userDetailId - represents which AccountDetail row belongs to a specific account
    - blackListId - represents Blacklist entity that belong to a specific account, can be null

- AccountDetails, extended account data is hold here. Data stored here is accessed less frequently so rather than having one single long Account entity is better to split in two separate entities with a one-to-one relation
    - accountDetailId - unique ID generated by DBMS engine
    - address – stores address of a user
    - title - holds the title of the user if any (engineer, doctor etc.)
    - education – stores education background of the users
    - experience – hold previous employment experiences if any

- Blacklist, whenever a HR Expert decides to ban a user a new record will be stored in this entity
    - blacklistId - unique ID generated by DBMS engine
    - reason – stores the reason why the user is banned from the system

- Advert, represent a job post. It has the following attributes
    - adCode - unique code for the job advertisement generated by DBMS engine
    - adHeader – brief description about the job
    - adDescription - full description of the job
    - adActivationDate - attribute used to make a job active in the near future
    - adDeadlineDate – attribute used to make a job inactive in the near future

- Application, represents an application made to an advert from an account(user). Has a many to one relation with both Account and Advert entity
    - adCode - composite primary key, references to an adCode in Advert entity
    - accountId - composite primary key, references to as accountId in Account entity
    - applicationTime - stores the timestamp whenever an application is made
    - status – current status of application

- Skill, represent a professional or personal skill
    - skillId - unique ID generated by DBMS engine
    - skillName - unique attribute containing name of a specific skill

- AccountSkills, entity that represents set of skills owned by an Account entity, has a many to one relation with Account entity
    - accountId – composite primary key, references to an accountId in Account entity
    - skillId - composite primary key, references to a skillId in Skill entity

- AdvertSkills, entity that represents set of skills required by an Advert entity, has a many to one relation with Advert entity
    - adCode – composite primary key, references to an adCode in Advert entity
    - skillId - composite primary key, references to a skillId in Skill entity

## 3.4 Data Flow Diagram

### 3.4.1 HR Expert Data Flow Diagram Level 1

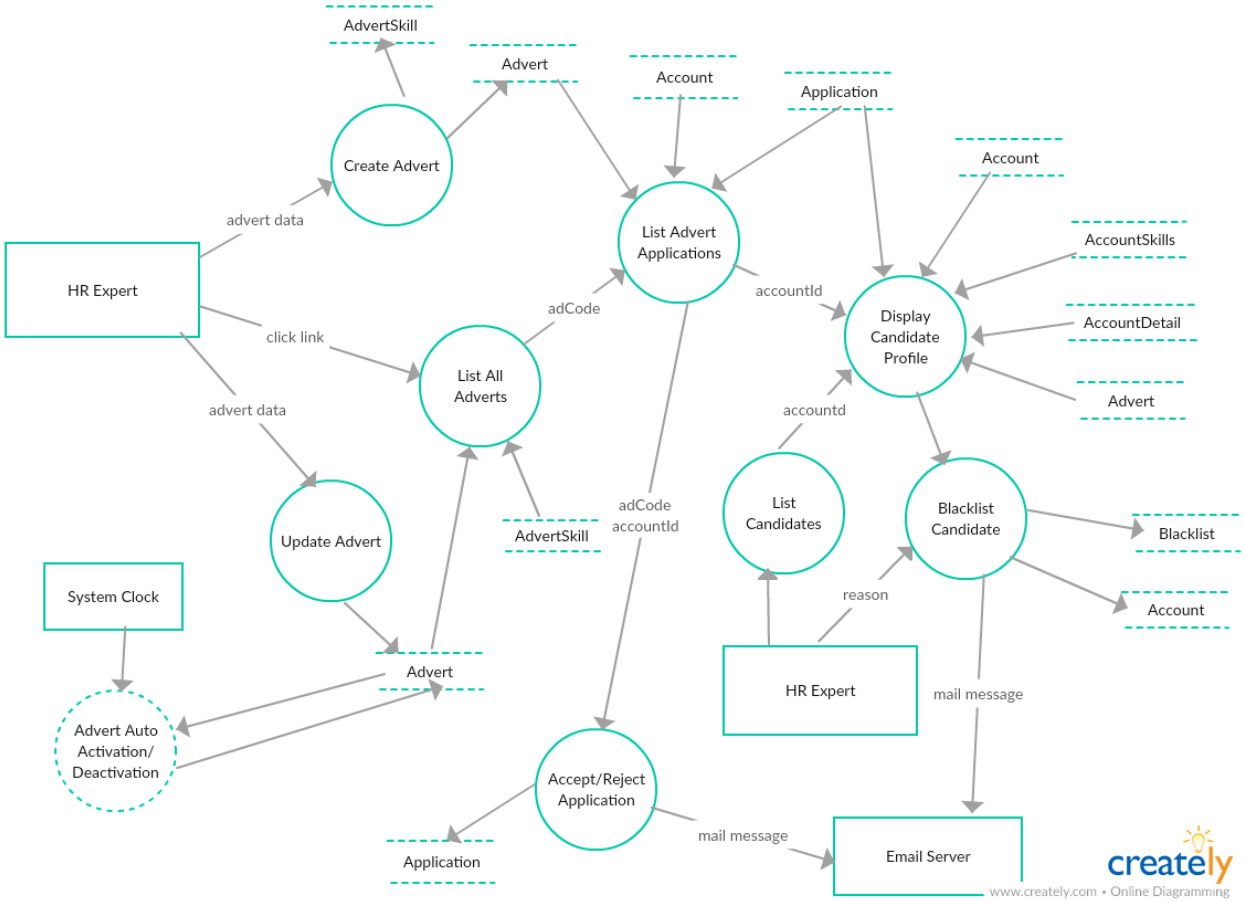



Figure 4: HR Expert DFD Level-1

I decided to break the system into processes that receive and transform data into another format or perform calculation.

*Create Advert:* This process takes data from HR Expert by using a form and saves it into Advert table and AdvertSkills table. Its responsibility is to create a new job advertisement.

*Update Advert:* This process (form view) takes an adCode from HR Expert, populates the form with the advertisement data and displays it to use for editing. On submit it updates both Advert and AdvertSkills table.

*List All Adverts:* Lists all adverts found in Advert table along with their AdvertSkills and displays them to HR expert for further processing.

*List Advert Application:* Shows all applications made to a job advertisement. Requires an adCode as input.

*Accept/Reject Application:* Rejects or accepts an application made by a candidate. Requires adCode and accountId as input. Output: updates Application table and sends a mail to candidate.

*List All Users:* No input required. Loads all users registered in the system (except administrators). Reads from Account table.

*Display Candidate Profile:* Shows a candidate profile along with all application he/she has made. Requires adCode as input.

*Blacklist candidate:* Bans a specific candidate from applying to any advert and rejects all his previous applications. Requires accountId and a reason String (text) as input. Output: Creates a new row in Blacklist table, updates Account table and Application table, forwards a message using email server to candidate.

*Advert Auto Activation/Deactivation:* An event process which listens to System clock. It is triggered every 24 hours. Input: Advert table rows. Output: Update Advert table rows by changing its status if conditions met.

### 3.4.2 User Data Flow Diagram Level 1

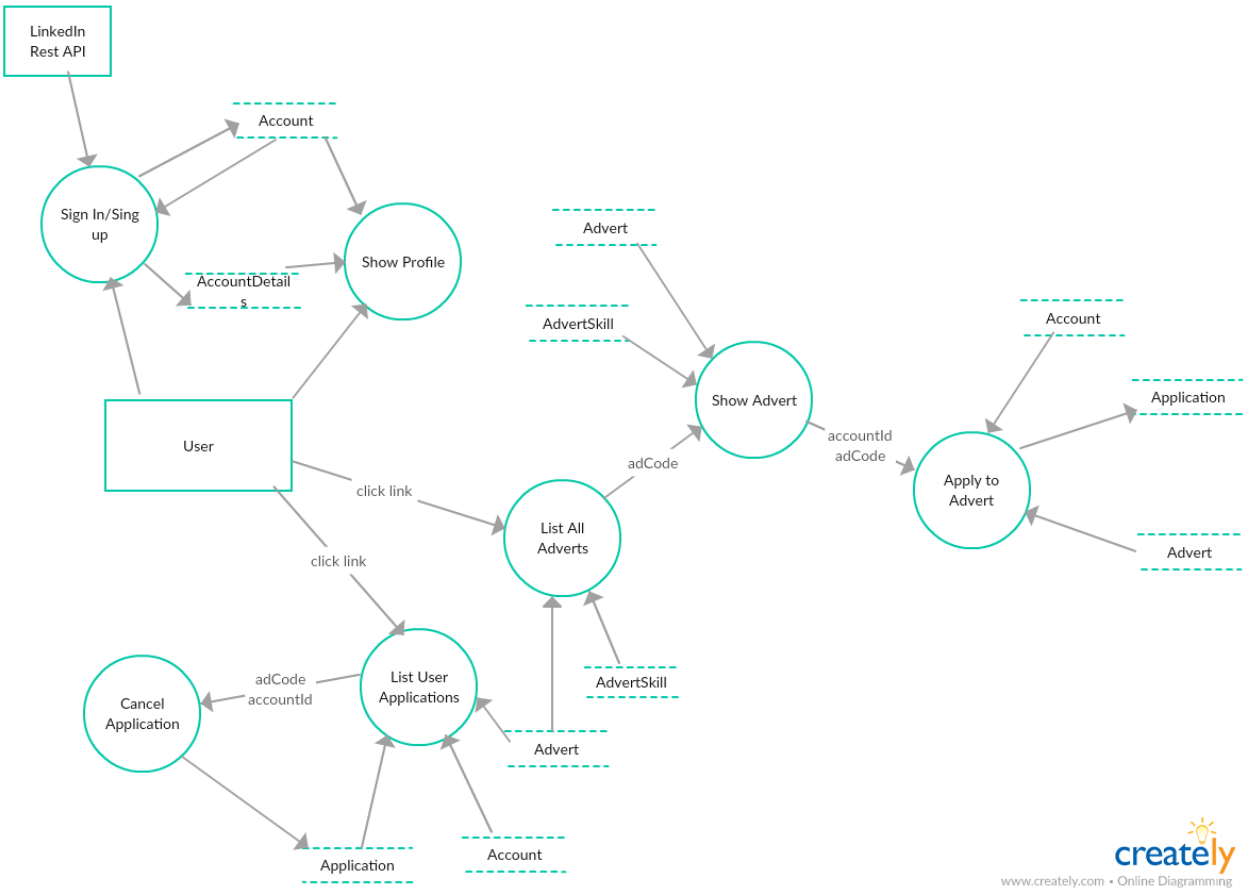



Figure 5 User DFD Level-1

*List All Adverts:* Displays a list of all active job adverts to both registered and anonymous users. Requires no input, it is triggered on link click. Reads Advert table data from database and displays it.

*Show Advert:* Displays a detailed description of a job advert to the user. Requires a adCode as input. Reads Advert and AdvertSkills from table and displays it.

*Apply to Advert:* Allows user to apply to a specific job advert. Requires adCode and accountId as input. Creates a new row in Application database as output.

*List User Applications:* Displays a list of all applications a user has made. Requires accountId as input. Reads from Application and Advert table.

*Cancel Application:* Allows user to cancel an application made to a job advertisement. Requires adCode and accountId as input. Updates Application table as output.

*Show Profile:* Displays detailed profile of a registered user. Requires no input. Triggered on link click. Reads from Account and AccountDetails table.

Sign in/Sing Up:  Allows user to sing into the system in order to apply to available jobs. Requires authentication data from LinkedIn as input. Creates a new row in Account and AccountDetails table if user isn't already registered. If user already registered, it only reads from Account table and compares data with credentials fetched from LinkedIn user data.