COMP 3004 - Deliverable #3
System Architecture and Design
`Brackit` - Mobile Tournament Bracket Creation

Metadata

Team / App Name: `Brackit`

Team member names

Name	Student ID
Jaime Herzog	101009321
Suohong Liu	101002340
Xiyi Liu	101004577
Alex Trostanovsky	100984702

Contents

Architecture

1 Description

In developing `Brackit`, we set out to address an urgent need by tournament attendants and organizers to visualize, manage, and interact with double elimination brackets on their mobile devices. At a high level, we committed to developing a product that will meet the following **functional requirements**:

1. Tournament Organizers (TO's) can create, host, maintain, and visualize double elimination brackets.
2. Registerd `Brackit` Users, as well as Guests, can use the application to join created tournaments.
3. `Brackit` will store and maintain user profiles that will describe users' history:
    - Matches won/lost
    - Tournaments entered/created

In terms of **non-functional requirements**, we believed `Brackit` be *usable* on mobile devices. `Brackit` users should be able to:

- View and access all submodules (Brackets, Rounds, Matches) of a tournament on an Android device
- Seamlessly enter tournament competitors to brackets on an Android device

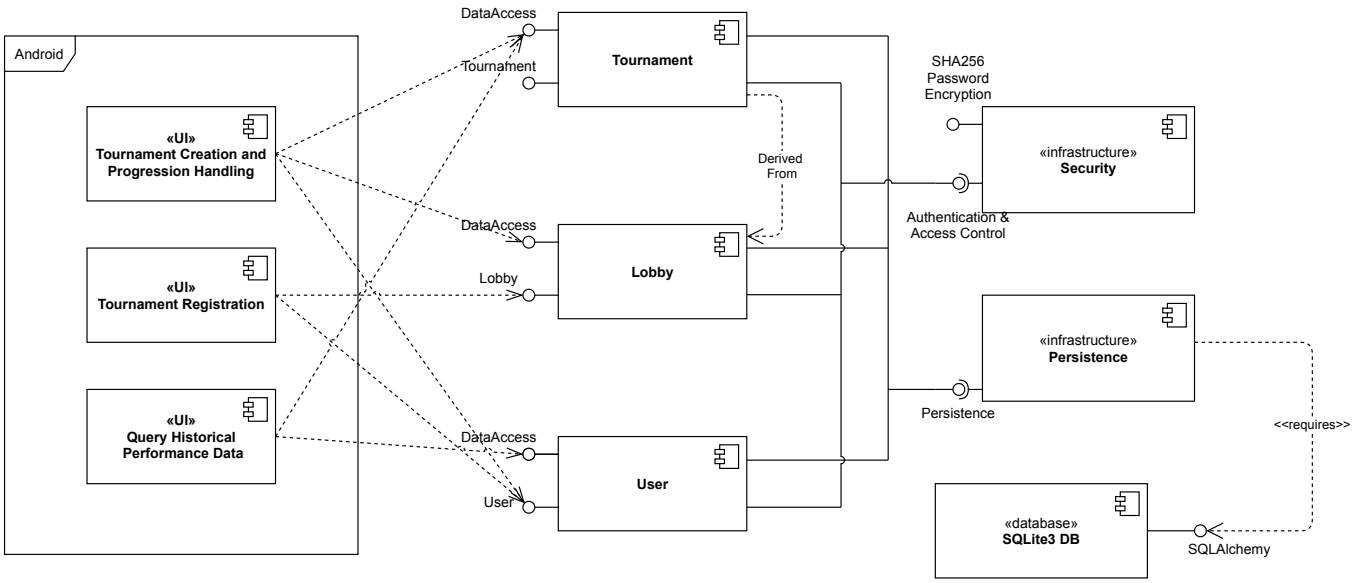
Conceptually, `Brackit` needed to support the creation and maintenance of the following *components*:

- *Tournament*: The highest level abstraction utilized in Bracket creation. A tournament acts a *container* for brackets.
- *Bracket*: Given the number of entrants and their corresponding seeds (ranks), Double elimination brackets dictate competitor matchups and the progression of competitors through the Winners and Losers brackets

2 Justification

To deve

3 Architectural Diagrams



Thanks to http://agilemodeling.com/artifacts/componentDiagram.htm

Figure 1: `Brackit` - UML 2 Architectural Component Diagram

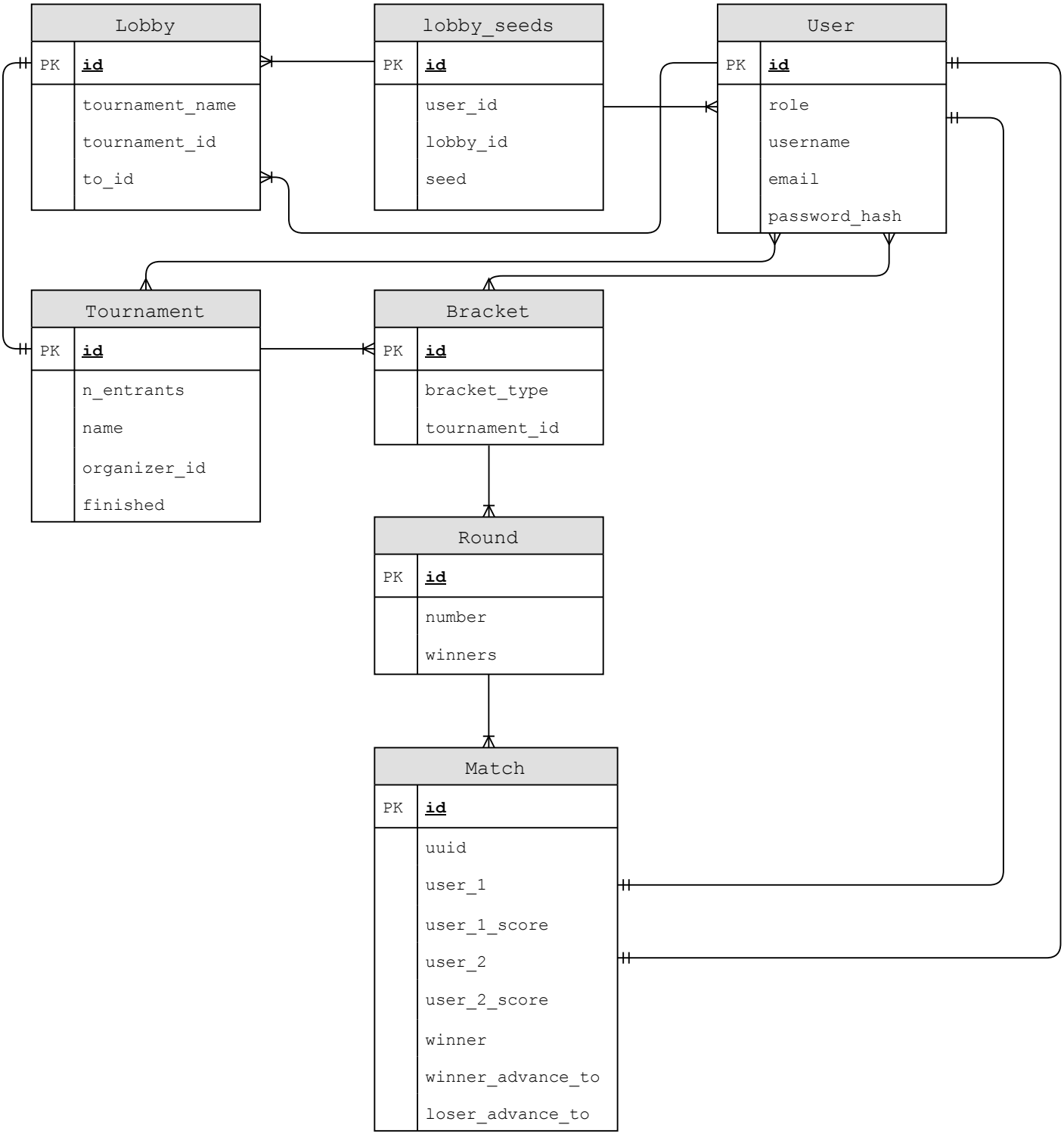


Figure 2: `Brackit` - Entity Relationship (ER) Diagram

Design

1 Description and Rationalization

- Use clear description of the structure of the components and its externally visible interfaces
- Clarify the physical location of where the classes will reside (e.g., on the client, on a server), as well as any external API
- Include references to your system's architecture (patterns, abstractions, data structures/ algorithms)
- An analysis of how your design minimizes coupling and accommodates changing requirements

Design Diagrams

Thanks to http://agilemodeling.com/artifacts/classDiagram.htm#CompositionAssociations

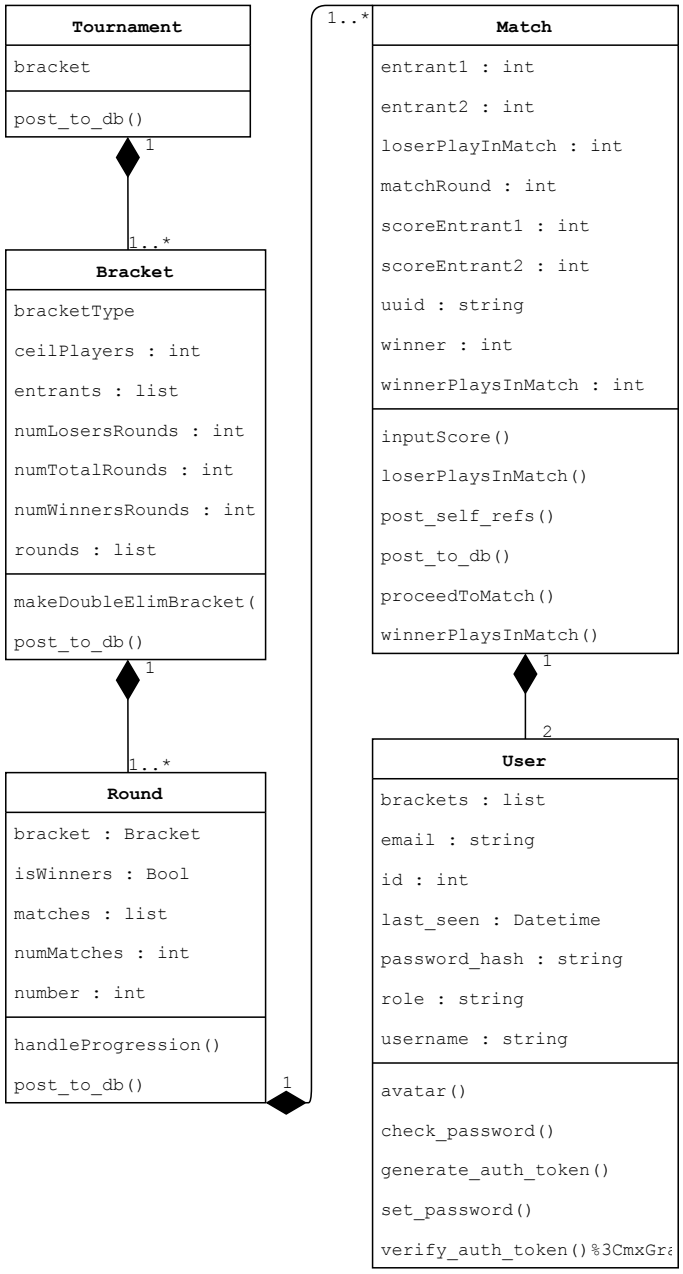


Figure 3: `Brackit` - UML Class Diagram

[1]

References

[1] Estimation lemma. Estimation lemma — Wikipedia, the free encyclopedia, 2010. [Online; accessed 29-September-2012].