Workplan

- Design (requirements, E-R model, schema's)
- System design
- Create of tables, views, database fill
- Implement functions and error checking
- Implement a graphical user
- Test, document and demonstrate

Report

Problem:
- application domain, goals
- user requirements & scenarios
- research: existing solutions, problems, differences

Approach:
- methods, techniques, tools
- planning, organization

Results:
- analysis: E-R model, functions
- database: table schemas
- system architecture
- GUI design

Evaluation:
- performance and usability tests
- db size, scaling
- possible extensions

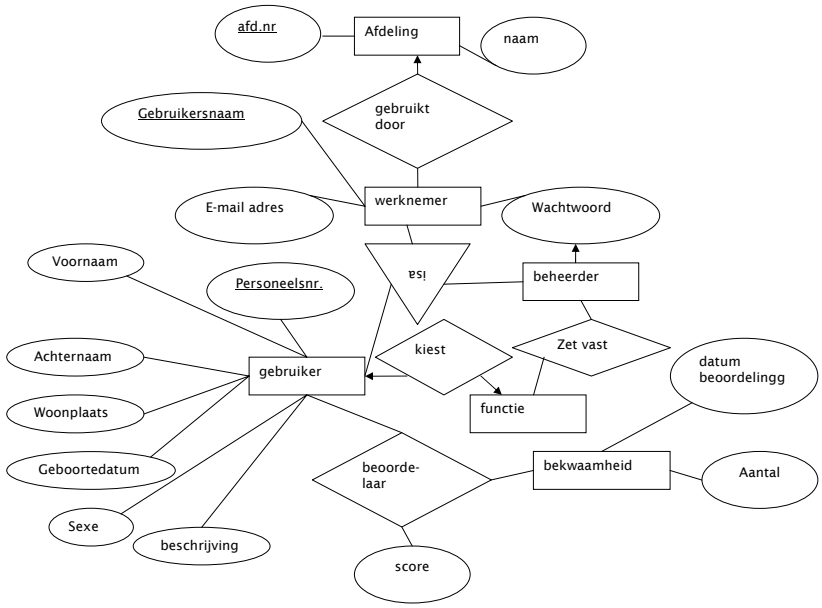
Report

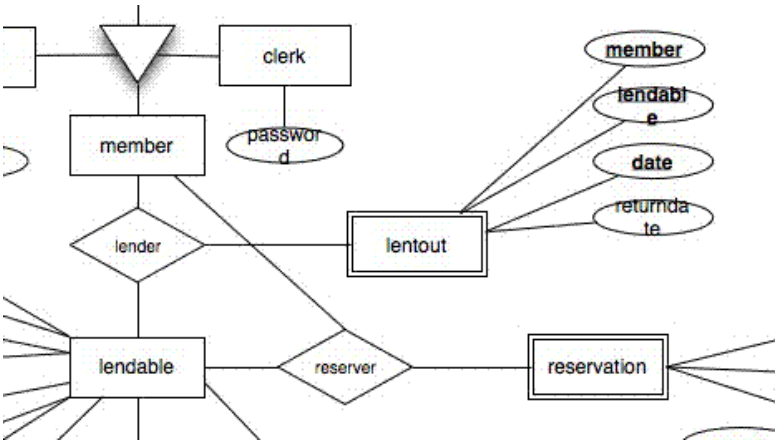
Appendixes:

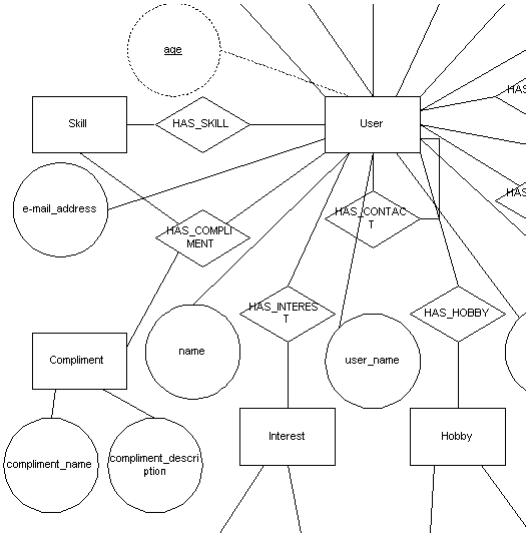
- user manual
- reference manual
- code listings
- table schemas
- case studies, test results

Do not forget:

- executive summary
- table of contents




clerk
member
password
lender
lentout
member
lendable
date
returndate
lendable
reserver
reservation


age
Skill
HAS_SKILL
User
e-mail_address
HAS_COMPLIMENT
HAS_CONTACT
HAS_INTEREST
HAS_HOBBY
Compliment
name
user_name
compliment_name
compliment_description
Interest
Hobby

Tools

MySQL *– See www.mysql.com in particular (dev.mysql.com)*

Databases on dbm.science.uva.nl:3306

JDBC *– JDBC the Java API for cross-DBMS connectivity to SQL databases and other tabular data sources, (spreadsheets)*

(See http://java.sun.com/products/jdbc/index.jsp).

MySQL Connector/J *- a JDBC driver to connect client Java applications to MySQL.*

See dev.mysql.com/doc/connector/j/en/java-connector.html

phpMyAdmin *-*

Step 1: load the driver that will handle the connections …..

```
import java.sql.*;
import java.sql.DriverManager;
import java.sql.SQLException;

[……..]

try {
        Class.forName("com.mysql.jdbc.Driver").newInstance();
    } catch (Exception ex) {
        System.err.println("Could not find driver");
}
```

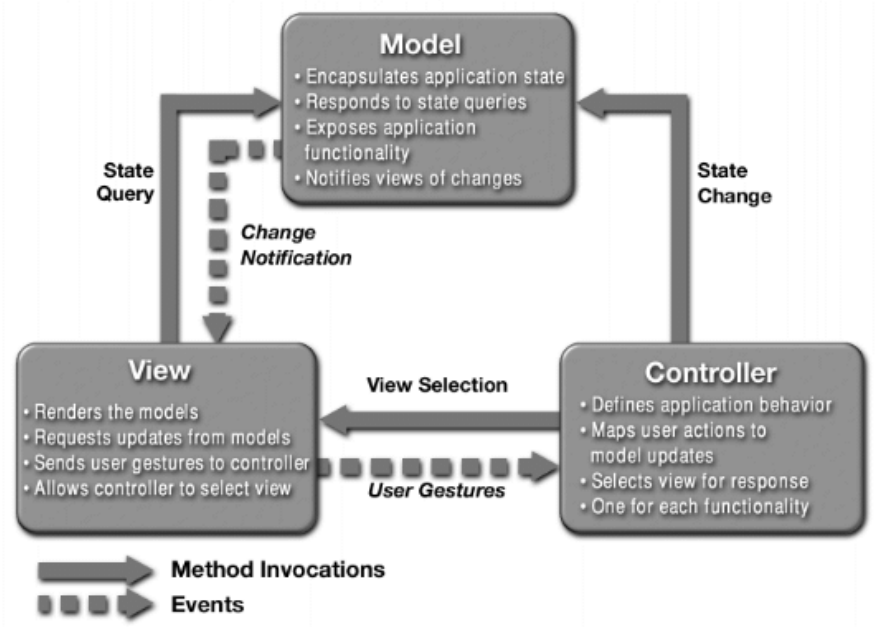
Step 2: get a connection to a database …

```
try {
    Connection conn = DriverManager.getConnection(
                "jdbc:mysql://dbm.science.uva.nl:3306/db",
"user", "password");

// Do something with the Connection

} catch (SQLException ex) {
        // handle any errors
        System.out.println("SQLException: " + ex.getMessage());
        System.out.println("SQLState: " + ex.getSQLState());
        System.out.println("VendorError: " + ex.getErrorCode());
    }
```

Step 3: do something with the Connection …

```
        Statement stmt = null;
        ResultSet rs = null;
        String q = "SELECT StudentName FROM Test";
        try {
            stmt = conn.createStatement();
                    if (stmt.execute(q)) {
                rs = stmt.getResultSet();
            }
            // Now do something with the ResultSet ....
          } finally {  // clean up by closing rs and stmt …
                  }
```

Model-View-Controller Pattern



Model-View-Controller Pattern

- Model contains data and business logic; it informs the view of changes
- Views show (explicit or derived) model content; they query the model
- Controllers handle user actions passed on by views, requesting updates to the model and/or the views; they will often correspond to the various functions