Software Design Document

Adriaan Leijnse
Version 1

1 Change history

Version	Comment
1	Iteration 1's basic design

2 Introduction

This Software Design Document describes the software architectural design decisions for the Xiast program. This document is geared towards anyone who might have to interact with the Xiast code base, by providing a high-level description of its implementation. It aims to follow the follow the IEEE Std 830-1998 "Standard for information Technology – Systems Design – Software Design Descriptions" standard.

Xiast is developed in an iterative fashion, starting with a minimal list of features to allow developers to familiarise themselves with the technologies used. Over four iterations the feature list will be filled out step by step, fully satisfying the Software Requirements Specifications. Please see the appropriate section in the "Software Project Management Plan" document for a description of each planned iteration.

This document will describe the design decisions for the current iteration, as well as directional pointers on how to proceed to the next iteration(s). It is a work in progress and needs further work to fully support the development team, testers and configuration manager in their work.

3 Definitions

JVM Java Virtual Machine [1]

Protocol A way to define interfaces in Clojure [2].

Docstring An in-code description of a function, interface or any other language feature supporting docstrings [3].

[1] `http://www.java.com`
[2] `http://clojure.org/protocols`
[3] `http://en.wikipedia.org/wiki/Docstring`

4 Context

The Xiast program is an HTTP server running a web application accessible via the user's browser. It's goal is to facilitate schedule management for universities.

4.1 Iteration 1

Use cases for the application are restricted to viewing student, course and room schedules. This means that the application has to be no more than a query interface for data in the application.

Viewing a student's schedule is restricted to authenticated users.

Only sample data is presented to the user, actual data will be integrated later via a database backend.

5 Composition of Xiast

Xiast's code is roughly subdivided in two parts: a back end and a front end. The back end includes the scheduler, persistent data management and features for managing communication between users. The front end interfaces with the back end and provides a web-based user interface. It does not contain any significant program logic, but caches schedules to allow off-line use of the application.

Xiast's back end is written in the Clojure programming language [4], which runs on the JVM. Front end code running in the browser will be written in JavaScript.

5.1 Iteration 1

In the first iteration the back-end and front-end are a single codebase: HTML for the interface is generated dynamically on the server and no code except Twitter Bootstrap [5] front-end UI framework's will run in the browser.

[4] `http://clojure.org`

[5] `http://getbootstrap.com`

5.2 Future iterations

Future iterations will have increasing amounts of code moved into the front end, reducing parts of Xiast into an online-accessible API, transmitting data via the edn data notation format [6].

6 Common abstractions in the Xiast code base

Xiast contains a number of abstractions and data types which allow code reuse throughout the program. Some of these data types will be used to transmit information from the back end to the front end and back.

6.1 Iteration 1

The interfaces and data types here are all designed to be extensible, ensuring that they can be used in further iterations.

6.1.1 The XiastQuery protocol

The `XiastQuery` protocol defines read-only access to Xiast's cachable data. This currently includes the course list, course schedules, student's personal schedules and room schedules. The full definition of the protocol is defined in the `query.clj` source file's docstrings.

The full definitions of the following data types can be found in the aforementioned source file:

Courses This data type describes a course, with its name and unambiguous identification string.

Schedule blocks Schedule blocks describe a time span on a specific day and location during which a class is taught, using the VUB's academic hours and calendar conventions. E.g.: Scheme is taught in week 3, on day 1, from 9am till 11am in room E1.03.

Timespans Describe a span of time over multiple days or weeks.

[6] `http://edn-format.org`

7 Persistent information

Xiast will need store various forms of information in order to function.

Fairly static facts about courses, their instructors, rooms, etc. will need to be stored, as well as dynamically generated user sessions, schedules and change requests between instructors and program managers.

7.1 Iteration 1

To allow rapid prototyping, the application only contains some mock data defined in the source code, loaded into memory at runtime, queryable via the Query protocol.

Student authentication is done through the VUB's authentication API; sessions are stored in encrypted cookies.

7.2 Future iterations

Future iterations will see the implementation of an SQL database schema to store updatable persistent information, as well as APIs to manipulate the stored data.

Users will also get their own persistent session data, as their settings are currently only retained when using the same browser.