Revision Planner Cross-Platform Web App

Georgi Koshov

4th Year Project Report
Artificial Intelligence and Computer Science
School of Informatics
University of Edinburgh

2014

Abstract

This is an example of `infthesis` style. The file `skeleton.tex` generates this document and can be used to get a "skeleton" for your thesis. The abstract should summarise your report and fit in the space on the first page. You may, of course, use any other software to write your report, as long as you follow the same style. That means: producing a title page as given here, and including a table of contents and bibliography.

Table of Contents

Chapter 1

Introduction

1.1 Application overview

The aim of the SmartRevise application is to allow a user to enter a list of subjects and the dates of their corresponding exams and have a revision schedule generated for them. Ideally the application should be accessible from a wide range of devices with varying screen-sizes and computational resources, schedules should be flexible - e.g. users can enter times when they are not available or change subject priorities, and all that has to be presented in a clear, aesthetic and easy to understand form.

There are several key problems that need to be solved in order to successfully achieve the conceptual aims of the project and to ensure that the minimal viable product could potentially survive and be successful in a commercial setting.

Application portability

With the introduction of powerful portable devices that achieve desktop-grade performance in terms of web browsing, it is no longer safe to make assumptions about the screen size or the browser of a given user. Therefore the design of SmartRevise has to be very agile and flexible: the interface elements need to have adequate size in case the user is navigating via touch, the interface has to be fluid, so that it makes the best of both desktop and mobile screens, the technology behind the application should use well adopted web-standards that have been integrated in the majority of modern browsers.

Solving the scheduling problem

Arguably the most critical task for successful completion of the project is developing a correct and efficient scheduling algorithm that is compact enough to be understood and discussed in its entirety. One might be tempted to implement a full-blown constraint satisfaction framework that allows the user to define sets of constraints imposing conditions on variables. A number of search techniques such as backtracking or local search could be used to derive a solution to a system of constraints on a finite domain. However I will argue in Section 3.4 that such solution is not required to successfully solve the problem as defined in this report. Furthermore constraint satisfaction algorithms

are inherently incomplete, that is an algorithm may solve a problem or prove that it is unsatisfiable, but there are some exceptions when a solution can never be derived. Therefore I have decided to focus on user experience, responsiveness and managed to develop an algorithm that is a fraction of the size of a traditional constraint satisfaction toolkit and is provably correct.

Integration with existing organisation tools

A lot of students use free tools such as Google Calendar or Apple Calendar to organise their schedules and have the information served to all digital devices they use. Google provides an API for developing applications using Calendar's infrastructure and the set up process required for a user to connect their Google Calendar with an external application is mostly effortless and arguably intuitive. Therefore on of the design goals for SmartRevise would be integration with Google's Calendar API and adding support for exporting the generated calendar events to a User's calendar. Additionally, creating a calendar experience within SmartRevise that resembles Google's will make the user's interaction less difficult and considerably more intuitive.

Chapter 2

Background

2.1 Scheduling

2.2 Constraint satisfaction

2.3 Existing applications

In order to identify potential pitfalls and set realistic standards for the success criteria of this project, two commercial revision-organiser applications were studied. Both GetRevising (http://www.getrevising.co.uk/) and ExamTime (htttp://www.examtime.com/) are browser-based web applications that feature organisation functionality for a number of activities typical for exam revision. The applications are free to use and require registration in order to create a revision plan.

ExamTime

ExamTime is the simpler of the two applications. After registering with the service, the user is prompted to add subjects to their dashboard. Subjects are in fact collections of mind maps, quizzes, flash cards and notes, organised under topics.

There is a calendar view where the user can manually add calendar events for exams, revision sessions or other activities. The application does not allow for the automatic generation of a revision schedule and the user has to complete this activity by themselves.

Get Revising

Get Revising offers a larger number of features such as Flashcards, Revision Notes, Quizzes, Mind maps and the ability to share resources between students. It also has some built-in layouts for common GCSE, A Level and University programmes that include subjects, known exam dates and simple content structure for well-known courses.

The user of the study planner feature of Get Revising can set School hours that should

be excluded from the revision schedule and the application has the ability to split the time prior to exam dates equally between selected subjects.

However Get Revising does not allow for a deeper level of detail when it comes to creating a schedule - only events for revising a particular subject can be created, therefore a user cannot include items such as*read textbook, solve tutorials, do past papers, etc.*

Chapter 3

Design

Due to the advancement of faster, more powerful web-browsers, featuring robust engines for running efficiently large amounts of code and with the continuos effort by organisations such as Mozilla and Chromium to drive web standards further and develop powerful open-source technologies for the next generation of Web, it is no longer infeasible to develop an entire application stack using a single programming language - ECMAScript, or more commonly known as JavaScript.

Some of the arguments in support of such design decision include the ability to exchange JavaScript objects between server and client without having to convert them during transport, being able to plug community libraries at all levels of the project without having to worry about compatibility or portability, but most of all due to the sheer convenience of having an entire application stack built with the exact same tools and following the same design philosophy.

3.1 Project architecture

SmartRevise is a client-side JavaScript application based on the Google AngularJS framework that is served by a NodeJS server. The entire application is downloaded by the client upon establishing connection with the server and the user's browser is responsible for rendering views and navigating between them. A similar approach is used by Google for their Gmail application and it allows for offloading the rendering responsibility from the server to the client. Additionally, after downloading the application, the only data that is exchanged with the server is in the form of JavaScript objects. Since the front-end can be cached by the browser, this solution can decrease traffic and improve the user's experience. On the other hand, since all further server requests are executed via an API to the database, it is trivial to implement memory caching, load-balancing and advanced protection to the leaner and less complicated back-end.

Application deployment, provisioning of the environment and dependency management is also handled by JavaScript services such as Grunt and Bower. Further details

about configuring and using those can be found in section 3.5.

3.2 Client-side application

3.3 Server and REST API

3.4 Algorithm

3.5 Additional tools

Chapter 4

Conclusion

Bibliography