Web Based Scheduling Utility

A Manuscript

Submitted to

the Department of Computer Science

and the Faculty of the

University of Wisconsin-La Crosse

La Crosse, Wisconsin

by

Jeffrey Powell

in Partial Fulfillment of the

Requirements for the Degree of

Master of Software Engineering

October, 2008

Web Based Scheduling Tool

By Jeffrey Powell

We recommend acceptance of this manuscript in partial fulfillment of this candidate's requirements for the degree of Master of Software Engineering in Computer Science. The candidate has completed the oral examination requirement of the capstone project for the degree.

_____	_____
Dr. Kenny Hunt Examination Committee Chairperson	Date
_____	_____
Dr. Tom Gendreau Examination Committee Member	Date
_____	_____
Dr. Mark Headington Examination Committee Member	Date

ABSTRACT

POWELL, JEFFREY, D., "Web Based Scheduling Tool", Master of Software Engineering, 09 2008, (Dr. Kenny Hunt, Dr. Tom Gendreau).

This manuscript describes a web based application capable of providing scheduling capabilities for multiple groups of individuals and shifts. It provides an environment in which department coordinators may create and modify schedules, and in which individuals in departments may specify their availabilities for scheduling. This tool was developed for a non-profit organization, but is generic enough to be easily adapted to any organization, and utilizes templates that more easily facilitate this adaptation.

ACKNOWLEDGEMENTS

I would like to thank my project advisors, Dr. Kenny Hunt and Dr. Tom Gendreau for their availability during this project. I would also like to thank the project sponsor, Living Word Christian Church, for recognizing a need and expressing interest in the potential for a solution. I would like to thank SAP Business Objects, Inc. for their support of academic advancement, as well as my coworkers and supervisors for their continued understanding of my unconventional schedule. Finally, I thank my wife for her ongoing patience throughout the duration of the Master of Software Engineering program.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

GLOSSARY

Access

Access is a desktop database tool developed by Microsoft. Within the context of this document, the sponsor organization had an existing database developed within Microsoft Access.

AJAX

AJAX, or Asynchronous JavaScript and XML, is a technology for use in web-based applications. It uses the browser to send requests for updates to the page, and then utilizes JavaScript to update the page itself, without any apparent reloading. This makes for a more seamless user experience.

ERD

An ERD, or Entity Relationship Diagram, depicts how individual objects relate to one another. Within this document, an ER diagram is used to represent the relationships between tables in the database used by the application.

Flash

A multimedia development platform developed by Adobe. It is frequently utilized for interactive web-based applications.

GUI

Graphical User Interface. The interface for an application as presented visually, as opposed to a purely textual interface.

Hash

A hash function takes data (in this case, the user's password), and performs some non-reversible procedure on that data to produce a fixed length string. Hash functions are, ideally, both simple and secure. A secure hash function is one for which it is difficult to identify the input from the output and difficult to identify multiple distinct inputs with the same output. By storing passwords as hash strings, it becomes infeasible for someone with access to the database to determine users' passwords.

MySQL

An open-source database implementation intended for use on web-servers. PHP has native support for MySQL.

ODBC

Open Database Connectivity, or ODBC, describes a standardized means of accessing databases. ODBC is intended to be independent of operating systems and languages, although the term ODBC is frequently used to refer specifically to Microsoft ODBC. Microsoft ODBC is a set of DLLs intended to allow a uniform means of connecting to various databases and datasets.

PHP

PHP, which is traditionally understood to refer to the recursive acronym "PHP: Hypertext Preprocessor", is a scripting language designed for use with dynamic web pages.

SSL

Secure Sockets Layer (and now Transport Layer Security) is a set of protocols for the encryption of data for transmission across the Internet. It is frequently utilized in online stores and in applications in which the unprotected transmission of private data would be undesirable.

1. Background Information

In a local church, there is, more often than not, a significant body of work that needs to be accomplished that many members of the congregation are completely unaware of. Some duties, such as garbage removal, the tuning of musical equipment, and the watering of plants, go largely unnoticed. Some tasks, like the staffing of a nursery or an ushering team, are significantly more visible. However, in many local churches, these positions are all occupied by volunteers.

While the funding available for paid, "professional" staff varies from church to church, it is often beneficial to make use of the individuals already present in the congregation. Volunteering provides individuals with a means to feel involved, connect with other members of the congregation, and contribute in a meaningful way. It allows a church a means to operate using people who are both already present and who want to participate.

However, as volunteers are not directly incentivized to be present, there is a certain amount of trust that a church must place in volunteers that they will be present to perform the duties that they volunteered for. Naturally, some people do a better job of this than others.

Those who tend to be more faithful in the tasks that are given to them also tend to be the individuals who are involved in multiple areas (either because departments recruit faithful workers, or because faithful workers want to be more involved). The result is that these workers tend to be scheduled more often across multiple departments. A single scheduling conflict can balloon into multiple scheduling issues. Swapping of shifts affects schedules in the future, as well as other departments active on the scheduled date.

For the sponsoring organization, as in many local churches, individual departments are independently coordinated by a single individual. These coordinators are responsible for producing schedules and distributing them to the people involved in their departments. As a result, a volunteer may receive multiple schedules for any given month. While individuals do provide information regarding their availability when they sign up for a

department, this information becomes outdated once they sign up for multiple departments or their obligations change. It also creates difficulties when coordinators schedule individuals at times when they have declared that they are unavailable.

Efforts have been made in the past to collate schedules together for departments existing in the same general area of service. Departments dealing with children have a group schedule. Technical departments tend to be scheduled together. This has had varying levels of success, as department coordinators have at times been inconsistent in providing schedules on time to identify conflicts before the final schedules must be printed. Regardless, insufficient communication between the individuals performing scheduling continues to result in conflicts for individuals serving. This particular issue was identified independently by multiple departments as an area where the investment of time and effort into a specialized tool for aiding the scheduling process could provide significant savings of both time and stress. The objective of this project is the production of a software tool that is capable of supporting the scheduling process in use at the sponsor organization.

2. Software Life Cycles

The different patterns by which software can be created are generally referred to as "software life cycles". They define the path through which a software project will travel from the time it is conceived until, in some cases, a planned obsolescence.

Software life cycle models have varying life spans and levels of meticulousness, but all attempt to produce the same result – a fully functional product that meets or exceeds the needs of the customer. These models range from the simplest case, that of no model whatsoever, to models more appropriate to large corporations and government agencies. The absence of a model would generally involve building the software simultaneously with generating the requirements, with no true design phase. This results in a prototype very quickly but also causes bugs and the oversight of possibly critical features, and can result in situations in which redevelopment from nothing is necessary. This approach is often useful for hobbyists or for proof-of-concept prototypes but lacks the documentation and rigor needed when developing more complex or mission critical projects.

At the other end of the spectrum, models exist that have been specially designed for very large software projects. These models involve multiple, often very lengthy phases. Requirements gathering, design, and review can take months or years, before any code is even written. While often inflexible and slow, the software produced is more likely to fit exactly what was requested and is likely to have undergone significantly more testing to ensure its correctness.

The developer chose a life cycle model for the sponsoring organization that combined aspects of both rapid prototyping and iterative development models. The sponsoring organization, having staff comprised mostly of volunteers, could not invest the large amounts of time necessary for some of the more involved life cycle models. As such, more rigid methodologies involving more meetings would serve to slow development more than would be acceptable. Additionally, due to the unique needs of some departments and the interactions among departments, identifying all possible features at the outset would be impractical. The concept of rapid prototyping would allow the

developer to better gather requirements and ascertain the needs and desires of the customer. By quickly providing a basic sample, the customer has the opportunity to respond to perceived and actual inconsistencies and shortfalls, as well as allowing the customer to realize that the original requirements were insufficient to accurately define the customer's needs. As part of this project, the initial goal was to have a model up and running as quickly as possible in order to get this feedback.

The iterative development model builds on the common waterfall model. In the traditional "waterfall" model, all progress flows downhill. In the unmodified state of this model, travel is one-way, with each step following the previous step and with no returning to prior phases at any point. The problem with this model is that, as development commences, more information invariably is obtained from the customer, allowing the requirements to be continually refined. The waterfall model effectively requires the developer to learn everything he or she possibly can and then figuratively locks him or her in a room until the product is complete.

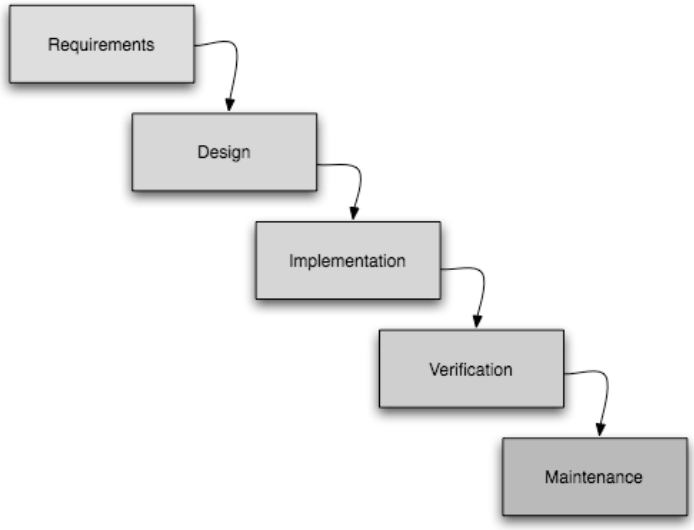


Figure 1. "Waterfall" software development model.[1]

[1] Figure reproduced with permission of Paul A. Hoadley

The iterative model, on the other hand, allows for the inherently cyclical nature of software development. After an initial planning period, the requirements gathering, design, implementation, and verification phases continue to occur until a product is developed that meets the actual customer requirements. During verification, the customer may identify critical features that are missing, which may result in the developer gathering more requirements, performing additional design work, and doing more implementation. This allows the developer to learn from the customer, from the previous iterations of the project, and from the testers. The use of this model also allows customers to regularly see progress being made and provide feedback, allowing for changes in design and scope without an overwhelming impact on the project timeline.

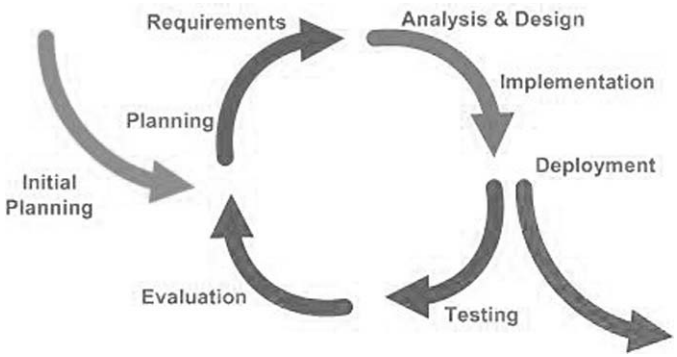


Figure 2. Iterative software development model.

3. Development Methodology

As stated earlier, department coordinators needed a means to create reliable schedules that were free of conflicts with other departments. In order to facilitate this, it was agreed that a system needed to be developed for all departments to use. Furthermore, as some needs were identified as being of low priority but suitable for inclusion at some future point, the developer determined that the code base should be developed in such a way as to make it easily extensible.

3.1 Initial Requirements

During early meetings with key stakeholders, the primary initial requirements were identified. As the life cycle model accounts for iterative development, the objective was to be thorough, although it was accepted that some requirements may be missed and need to be added later. An additional benefit was that the developer started with a considerable understanding of the application domain. The majority of the staff interviewed were unfamiliar with common software terminology (even, to a large extent, basic computer terminology), which resulted in the requirements being distilled into a more non-technical (and oftentimes conversational) form. This assisted, in many respects, the development process, as it required that requirements and concepts be examined as a non-technical user would.

Initial requirements were gathered over a period of roughly one month by meeting individually with five stakeholders. Meetings themselves were informal and often consisted of a discussion of the concept of the tool along with questions and answers regarding issues expected in development, areas that were determined to be unclear, and areas in which different features presented themselves. As these meetings were informal, approval of requirements often consisted of individuals showing excitement at seeing a potential solution to their scheduling difficulties. This is not to suggest that those involved in reviewing requirements and prototypes were not helpful, as they provided invaluable insight into the use of the tool. Rather, at times, possibly troublesome areas of

the application did not receive as much scrutiny as they would have were the reviewer perhaps more technically inclined.

As a starting point for discussions regarding requirements, some ideas were generated regarding what could most benefit the scheduling process. These were presented, at first, to an office worker with some responsibilities for the consolidation of schedules, as well as with scheduling responsibilities of her own. With her support, and the support of the office manager, a draft of the critical features was produced. These were discussed with two childcare coordinators. The individuals chosen represented a cross-section of the scheduling responsibilities at the organization, and each had a vested interest in the reduction of scheduling conflicts. Following these meetings, the developer synthesized the requirements in Table 1 from the issues and ideas discussed.

1	Currently, schedules are created on paper. The tool must be capable of removing the requirement for paper-based scheduling.
2	The system must be accessible via the local network or the Internet. Individuals must be capable of logging on to the system from home.
3	Often, individuals are scheduled in multiple locations simultaneously. The application must actively prevent this. Individuals creating schedules must not be permitted to create a schedule with an individual that is already scheduled.
4	Currently, individuals specify their availability for scheduling by providing a list of how much they want to be scheduled during a given month. The system must support this method of stating availability.
5	The system must allow users to print their schedules.
6	The system must allow departments to print schedules.
7	The back-end database must support being connected to the existing church database so that data is consistent across all sources. There must not be the need to regularly enter new data into the church database as a result of updates to the application database.

8	The system must meet the current needs, but must not be limited to them. Moving forward, this system must be capable of adapting to changing requirements.

Table 1. Initial Functional Requirements.

1	Any online databases must be available via the existing website hosting plan.
2	Any dynamic page generation must be compatible with the existing website hosting plan.
3	The system must be inexpensive. Much software has already been purchased, and no additional expensive software is wanted.

Table 2. Initial Non-functional Requirements

3.2 Initial Design and Prototype

The first prototype was started in November of 2007. This prototype provided a limited set of features that demonstrated the tool's navigational setup and future functionality. Where functionality had not been implemented, placeholder graphics were present to provide some context regarding the intent of the page. While visually unappealing, it demonstrated that the initial interpretation of the requirements was reasonably accurate. Use of this prototype allowed for the improvement of existing requirements and the addition of new requirements.

Initial GUI design consisted of producing a set of hand-drawn mockups for review by the childcare staff. This allowed for the quick determination of what buttons would be present on which GUI screens, where they would lead, and what they would accomplish. Following the approval of these rough designs by the childcare staff, development of the first prototype began. Internally, a design document was drafted, although not presented to stakeholders. This document, along with the first prototype, acted as a first iteration through the cycle in the life cycle model (see Figure 2 and accompanying text).

Stakeholders could see a prototype and identify requirements that had not been identified yet.

At the time of this undertaking, the developer had limited experience with PHP as a web development language. However, the developer determined that as the tool needed to be accessible via the Internet, the best implementation was a purely web-based solution. As a result, a language suitable for web-based application development was necessary, and PHP was chosen as the most appropriate given the organization's web hosting plan's support for it as well as the large body of community assistance available. The amount of freely available support for competing web-development languages is significantly more limited. Furthermore, PHP itself is free, while having the same major capabilities as other web scripting languages.

Based on the website hosting that was in use by the sponsor organization at the time, the database chosen to support the tool was MySQL. MySQL provides support for interfacing with Microsoft Access, which would allow connection to the existing database, as had initially been planned. As with PHP, MySQL is freely available and is installed with most web hosting packages. Aside from annual hosting fees, the use of PHP in conjunction with MySQL provides a zero-cost solution.

3.3 Scope Changes

During the development of the second prototype, previously unforeseen circumstances brought into question the feasibility of functional requirement number seven (see Table 1). While MySQL does support an ODBC connection to Microsoft Access, the hosting company for the existing website refused to provide an open port to allow the existing office database to communicate with the scheduling database. In light of this, the developer opted to expand the scope of the database to support a subset of the functionality available in the database in use by the office staff.

The lack of an interface between MySQL and the existing Access database meant that some of the functionality present in the existing office database needed to be provided in the system. After examining the requirements gathered thus far, it was estimated by the

developer that, by omitting the requirement for automatic schedule generation, while retaining all features and functionality for manual scheduling, the implementation would remain roughly on schedule for delivery by the end of the Spring 2008 semester. As manual scheduling was identified as the most likely use of the application in the first place, and after brief discussion with one of the childcare staff members, the developer decided to remove the automatic schedule generation functionality, as long as the core scheduling functionality and support were still provided.

An additional major change occurred when, while previewing the second prototype, one of the childcare staff identified a usability shortcoming in the GUI that required an overhaul of the GUI, as well as how a number of different modules interacted. The tool, as it existed, guaranteed that an individual could not have a scheduling conflict. However, she provided some specific examples of acceptable scenarios in which an individual will have multiple concurrent scheduled responsibilities. After working with her, providing samples of new GUI screens and identifying solutions to rectify the incorrect scheduling constraints, she approved the changes to the GUI, although a significant amount of developer time was spent reworking these issues.

3.4 Iterative Development

Throughout the development of the application, one of the developer's goals was to produce a tool that was modular in nature. It was anticipated that future development work would also be done for the organization on this tool, so the developer determined it to be in his interests to produce a tool that would be easily extensible. By producing a tool that was modular, it was significantly easier to iteratively produce prototypes. With functionality in relatively self-contained modules, specific items could be included on pages as needed. User authentication, for example, existed within a single source file, which was included at the head of those modules that required the features available in that source. This was adhered to wherever possible to prevent redundancy within the application source code, and it applied both to core functionality (such as authentication, menus, and layout) and to specific feature areas (such as scheduling, user profiles, and

reports). At the same time, this made development more iterative in nature – individual modules could be created and tested one at a time.

Following the first and second prototypes, demos were given to the childcare staff members. Additionally, following the second prototype, the office workers were given a demonstration as well. These demonstrations provided feedback regarding how features were being presented to the user, as well as feedback regarding refinements necessary within the GUI itself.

3.5 Final Design

The application consists of a number of key components: the interface, which exists client-side in the web browser; the database back-end; and the executable code, which exists server-side and which, in executing, provides the interface as necessary.

High Level Architecture

The application exists as a series of PHP pages. As PHP is rendered server-side, this allows the application itself to be hosted on the organization's local network or on an Internet server, as necessary. The server either contains or interacts with a MySQL database housing all the data necessary for the application. Depending upon the actions of the user, the server provides content appropriate for the requests that the user has made.

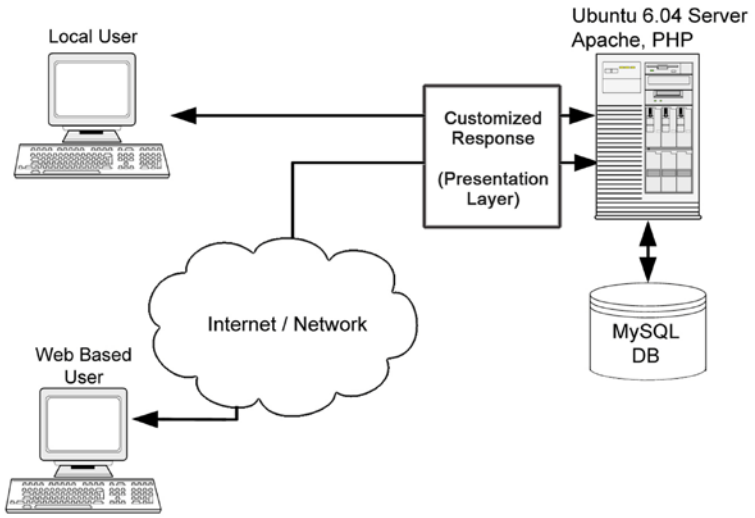


Figure 3. High level architecture.

.

Detailed Architecture

The core application exists as a dynamic page, loading content specific to that requested by the user. In Figure 4 we see the user's home page with dynamic content areas outlined. The layout of every page is similar, but the content depends upon the selections the user makes. Along the top of the application is a navigation menu. If the user is an administrator, an administrative menu is present. The left-hand sidebar acts as a context menu area. If the user requests a page having content with context sensitive options available, the options will appear in the sidebar.

The main page imports the custom content as it is needed. If the user has opted to go to the home page, the template for the "home" page is imported. Within the home page template itself, individual content objects may exist and may import other aspects of the application as necessary.

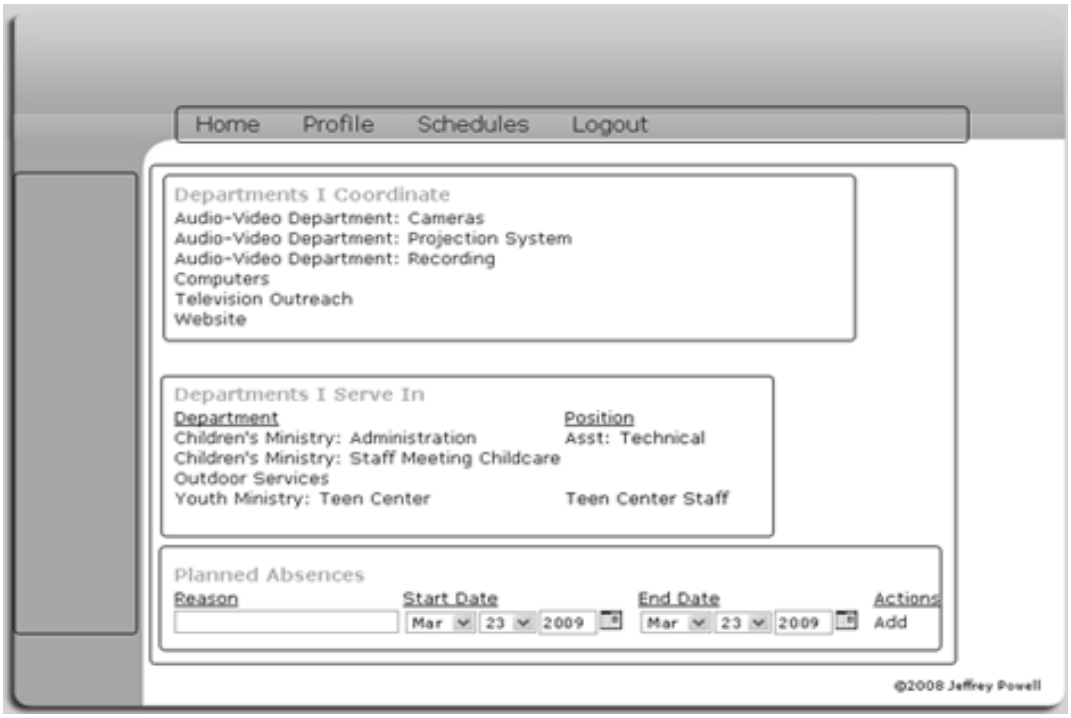


Figure 4. Dynamic page content example

The available menu options and associated page templates depend upon the modules that are present. Once the main page is aware of other modules and how to treat them, they are presented along with the rest of the content.

Modules, as they are imported into the main page, inherit access to many of the core functionalities provided by the application. This makes development of new modules a reasonably efficient activity. Issues such as dealing with security, database access, and schedule handling are abstracted by the application itself. The primary modules available at this time are described in Table 3.

Module	**Functionality**
Core	While not technically a "module", this consists of the index page, primary includes, and originally defined classes. As such, it provides the basis for other modules by providing login, logout, page generation, database access, and security.

Home	Provides an overview of key information for users and unites aspects of different modules. Synopsizes involvement in all departments and allows creation of absences.
Profile	Provides access to view and edit database information stored about the logged in user and allows setting preferences on the privacy of stored data.
Department	Provides a concise view of a given department for individuals within that department. Coordinators are provided with the means to specify which shifts are worked, along with a view of the staff within the department. Others are provided with their schedule and the ability to specify their availability in that department.
Schedules	Provides the means for department coordinators to create and edit schedules, and also allows users to view their schedules, both on a department-by-department and on an organization-wide level.
Reports	Allows office staff to generate pre-designed reports.
Administration: Shifts	Allows the creation, editing, and deletion of any shifts.
Administration: Personnel	Allows the creation and removal of individuals within the application. It acts as a user-management page for administrators.
Administration: Departments	Allows creation and removal of departments within the application. As with the personnel module, it is effectively a management page.

Table 3. Modules available for application.

Each of these modules is designed for one or more classes of user. As such, the functionality can change depending upon the logged in user.

Class Diagram

As PHP has many object-oriented aspects, the creation and use of classes yields a better design in many areas. While, as a web-based application, a great deal of code

relating to core functionalities exists at the page level, it is reasonable to view each page itself as a class, with the functionality it needs built in. An example of this can be seen in Figure 7 – while classes are not as thoroughly defined as they might be in a traditional application, PHP allows the use of class-like objects to make the code cleaner and the application easier to maintain.

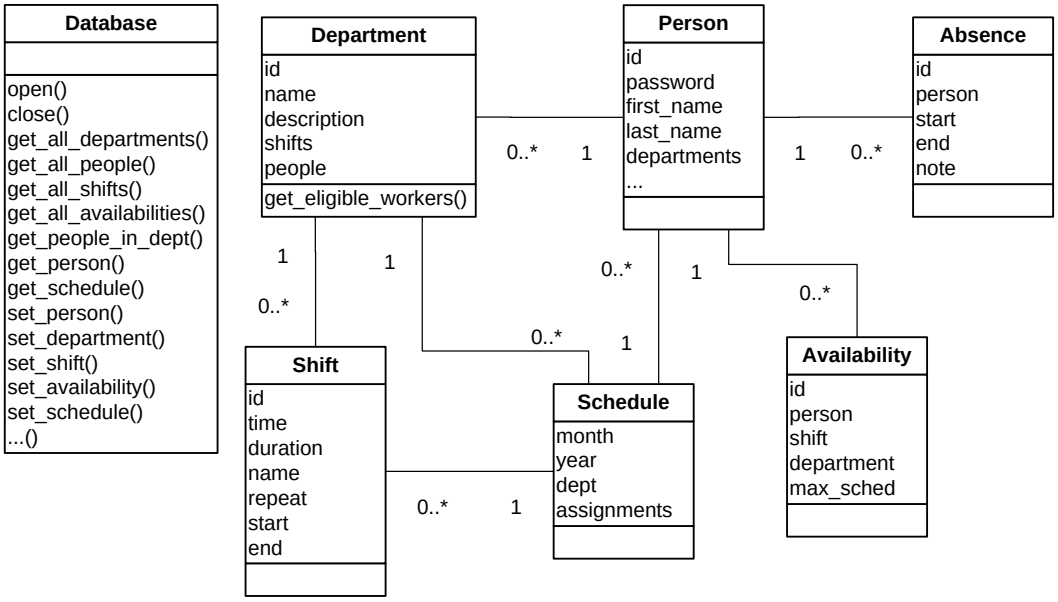


Figure 5. Basic Class Diagram

Database Design

The database is the backbone of the system. It contains all the data necessary for the application. Following the change in scope, it additionally includes the existing church database. As such, while database security is not presently a major concern, it is a legitimate consideration for future development. While the nature of the current data set is not such that it would be considered too risky if users were to have access, the intent was that it be possible to easily support more mission-critical applications using the same code. As such, only administrators have direct access to the database, and critical information moving between the client and the database has the capacity of being encrypted or sent via SSL.

The database itself has undergone a few schema changes. The change in scope allowed for a consolidation of some pieces of data. Much of the existing office data was permitted to aggregate into the system database, with minimal impact to the functionality of the system itself.

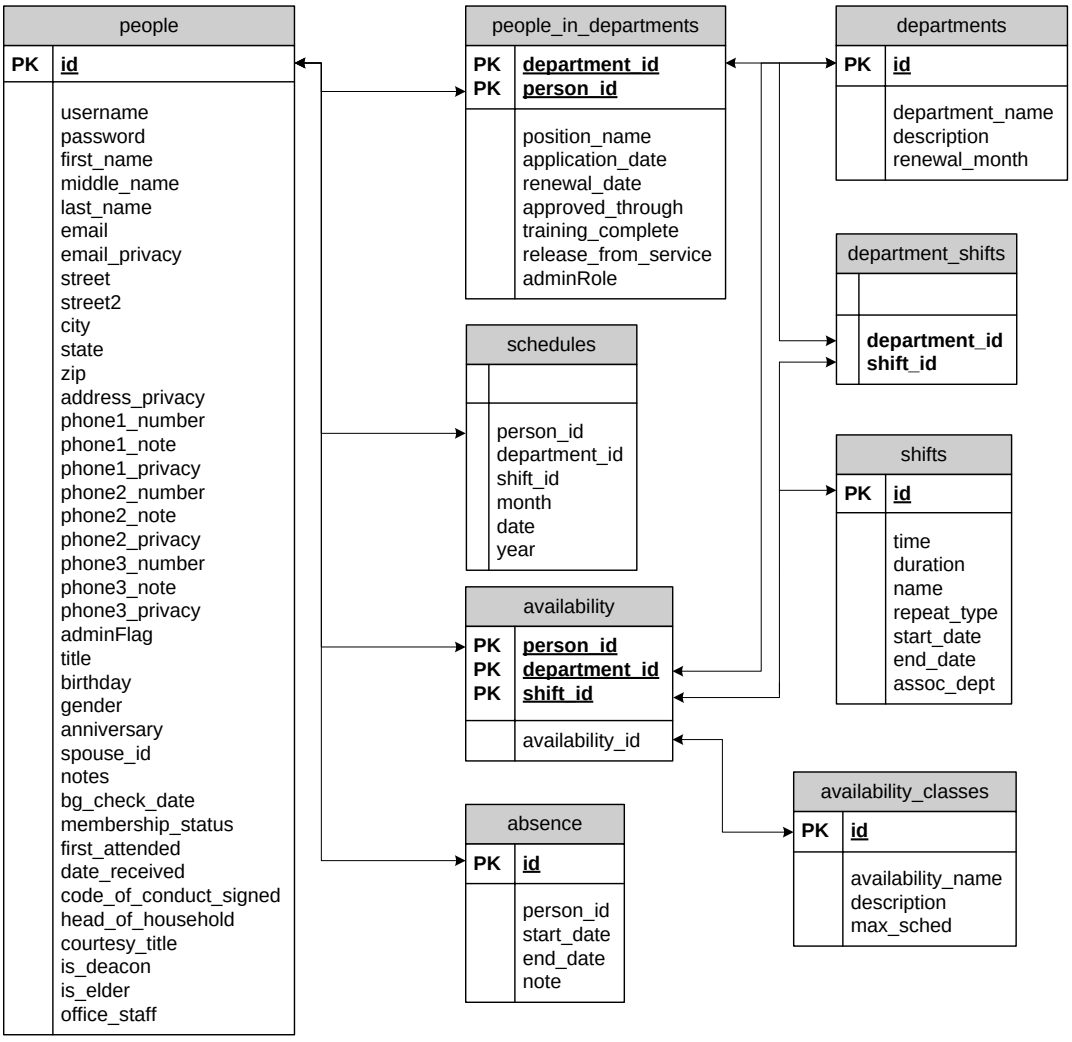


Figure 6. Entity Relationship Diagram

The "people" table consists of records relating to each individual in the organization. Some of the information is editable by users within the application. Other aspects of the information are viewable only to office staff. The existing office database included much of the contact information, which was preserved within the system. Also maintained

were additional characteristics relating to the specifics of individuals becoming involved in departments within the organization.

Similarly, the “departments” table describes the nature of each department in the system, with the “people_in_departments” table providing the intersection point for individuals serving in a given department, indicating who is in which department and what his or her role is within that department.

The “shifts” and “department_shifts” tables define the periods during which a given department is active. In the example of the sponsor organization, a church, many departments are active during the period in which a service is going on. Some departments, such as the cleaning staff, may be active only when services are not occurring.

A user is capable of defining his or her availability and any temporary absences within the system. This data is stored in the “availability” and “absence” tables. These tables describe how an individual should be scheduled and dates on which no scheduling can occur. The types of availability are defined in the “availability_classes” table.

The core of the scheduling application functionality is the “schedules” table. This table stores information relating to the “who, what, when, and where” of the schedule.

All these tables work together to represent the personnel present at the organization, their involvements in the various departments, and how the departments make use of their availabilities for scheduling purposes. With the additional information present from the preexisting database, some future design options present themselves as well.

Interface Design

The interface consists of a number of pages that are built dynamically by the application. The standard output of PHP is to the browser as HTML. As such, creating a page using PHP allows complicated dynamic content, while the end user sees only a page of resultant data, not the executable code. This gives the impression of a “custom generated” page, specific for the logged in user.

The interface contains a navigational menu, a context menu, and a content area, with the content area further subdivided into content objects. All of these items are imported into the page, depending upon the user's actions. For example, selecting "Schedules" on the navigation menu will show the current schedule object in the content area, leave the menu the same, and provide context options associated with viewing a schedule.

Graphically, the user interface was designed to be unobtrusive with minimal graphics. A masthead area is present for an organization logo or other information, but this area can be minimized with negligible effort within the main template.

Navigating within the application occurs by way of the navigational menu and its sub-menus and the context menu (when available). In the example in Figure 7, the user is viewing a schedule object. Users have access to the menus in the navigational bar, allowing them to view a different section of the application, or they may use the context menu along the side. The context menu provides them with access to the schedules associated with different departments in which they are involved.

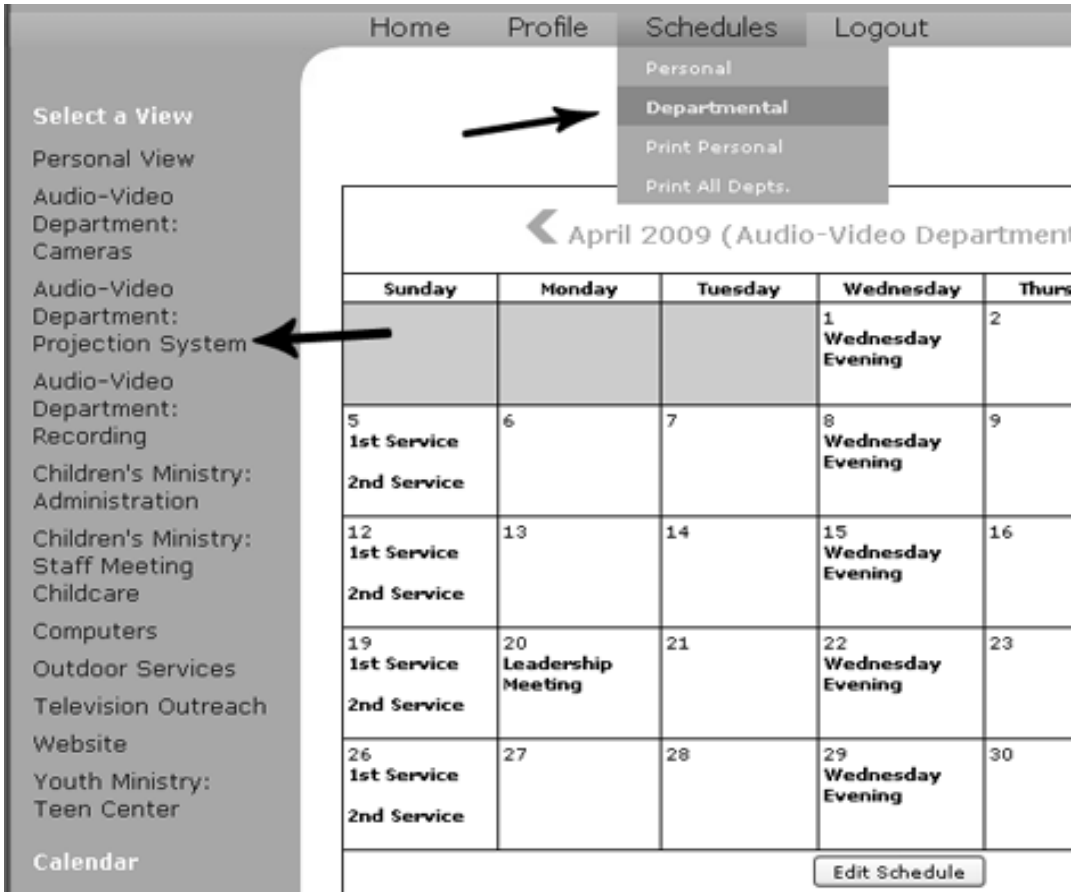


Figure 7. Navigation methods within application.

Security Considerations

With the expectation of privacy, there must be an appropriate means of providing secure interfacing with the application, as well as ensuring that users are not permitted to see information that would be considered private by other users. This is accomplished in a number of ways.

Logging in is accomplished by means of a username and password. On the database itself, the password is hashed, preventing a database administrator from viewing login information. This being said, it is assumed that an individual having direct access to the database also has the permission necessary to view any of the contents of the database. That is, there is no wholesale encryption of the data.

From a user's profile, it is possible to declare one's contact details to be either public, visible only to coordinators of departments in which one is involved, or private (visible only to office staff). This capacity allows users to easily define whom they want to grant access to based upon whom they expect will need access to their information. Furthermore, as the data that is present in the user profile is directly linked to the organization's database, it is possible to remove contact details that the user does not wish to be present. It should be noted that this database is not linked to the financials database, and any information provided with charitable giving is still associated with the individual, while this database may have limited or no information on the individual.

3.6 Testing and Deployment

During the early stages of development, testing was performed by the developer. The creation of some key use cases allowed walking through major functional requirement areas in order to identify possible problems that users might experience.

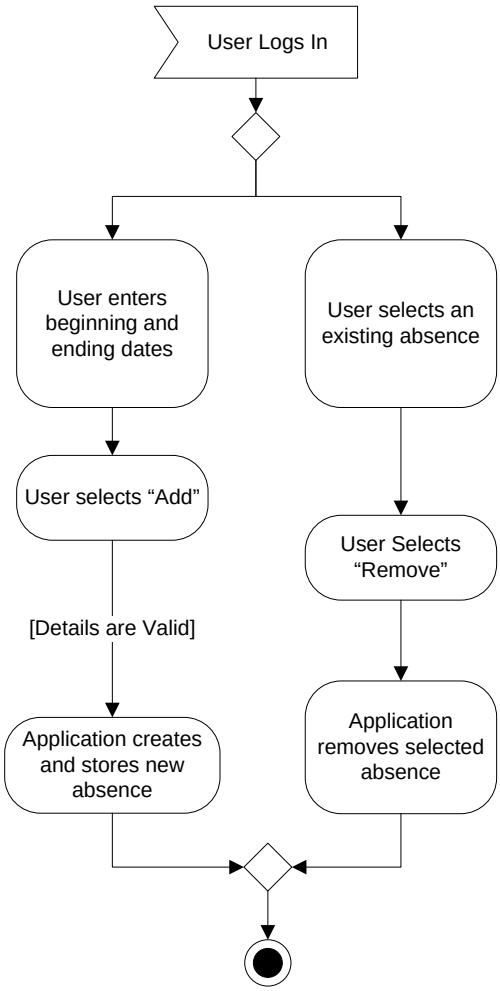


Figure 8. Activity Diagram: User Creates or Modifies an Absence

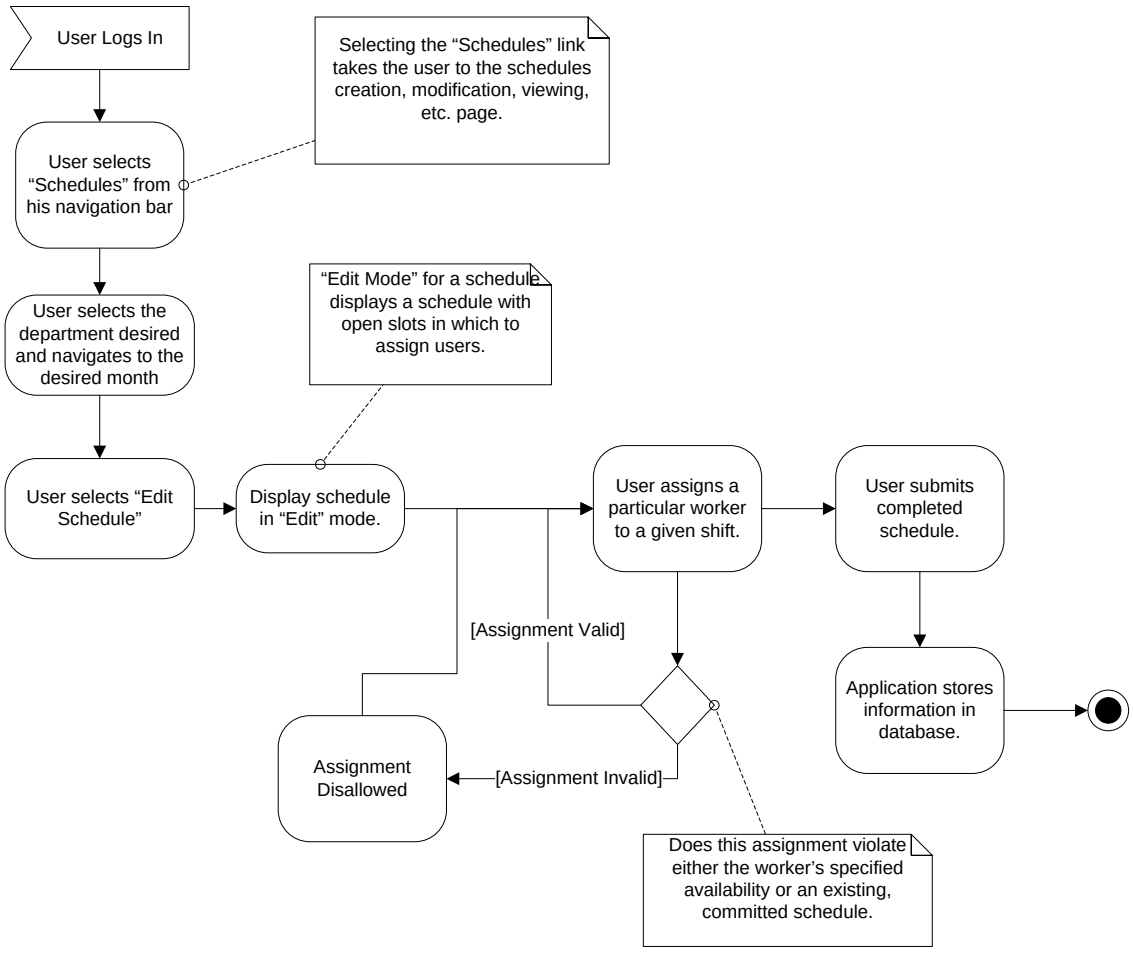


Figure 9. Activity Diagram: Department Coordinator Modifies a Schedule

Figure 8 depicts how a user would go about indicating an absence (during which he or she could not be scheduled). Testing based upon this identified bugs in the manner in which absences were stored in the database. Figure 9 represents how a department coordinator would create a schedule. In testing the process presented, weaknesses in the presentation and selection of volunteers in a schedule were identified. Similarly, the activity diagrams in Appendix A led to a number of bug fixes, as well as improvements unrelated to bugs.

While the activity diagrams served to identify bugs in the usability and functionality, they failed to address the issue of training. As the developer was fully aware of how the

application “should” work, this testing did not sufficiently test cases in which the user is unfamiliar with the tool and not performing actions in an expected manner.

During later portions of development, a childcare coordinator and the coordinator of another department were used to test the application. With some guidance, they were walked through activities that they would typically perform outside the system. This activity also served as a means to provide an individual that had not yet seen the application a chance to see what was possible, effectively providing additional buy-in.

Following meetings with those involved in forming the original requirements, it was determined that the tool, as presented as the final iteration of the second prototype, met the requirements. As such, deployment began in September of 2008. The application was initially rolled out to those departments responsible for childcare to perform their scheduling in parallel with their existing scheduling. These departments historically have had a significant problem with scheduling conflicts and serve to gain a great deal by the introduction of this tool. As such, they were also identified as being more likely to want to try the tool.

Unfortunately, as of February 2009, the tool has yet to enter usage organization-wide. Although the tool was presented to the childcare departments, the office staff, and other individuals responsible for scheduling, the developer failed to receive any feedback on any usage difficulties or, for that matter, any information whatsoever from the initial deployment group.

At this time, a second deployment group has been identified for which the developer has direct oversight. Those responsible for scheduling in the second group will, instead of being provided access to the tool and requested to evaluate it at their leisure, be personally walked through use cases and have the use of the tool mandated.

While the requirements themselves have been satisfied, due to the relationship of the developer to the sponsor organization, there is some expectation on the part of the developer that, as deployment continues, a more involved maintenance period will occur, with the addition of new features and the modification of existing features. As any

additional modifications are integrated, the deployment will, ideally, be expanded to include all departments. Additionally, as the tool supports permitting non-scheduling users within departments to provide additional information regarding their availability and any anticipated conflicts, the deployment will eventually be expanded to include all individuals within all departments. At that point, deployment will be complete, and all office systems, policies, and procedures will have been switched over to make full use of the new application. At the same time, ongoing maintenance would begin.

4. Limitations

The application meets the functional requirements and works well for its intended purpose. In that respect, it should not be considered limited. However, there are features that "power users" may like to see but which have not been implemented. While not required, they were identified during development either by the developer or by staff at the sponsor organization as features that would improve the tool. Some examples of these limitations are:

Table 4. Application Limitations

1	Users do not have the means to use the system to seek replacements. As a result, after finding a replacement when an individual is unable to work during a given shift, the schedule continues to show the originally scheduled individual.
2	Schedules currently have a simple drop-down style for selecting individuals to schedule. Unfortunately, for those departments that regularly schedule multiple individuals per shift, the method for adding additional scheduling slots is cumbersome.
3	Currently, there is no provision for multiple distinct positions for a single shift within a given department. While a department may, for example, schedule multiple teachers in a classroom, it may not specify that one teacher is a lead teacher and the other is the assistant.
4	The database stores the ID of the spouse of an individual. While this was present in the existing church database and is not required, its presence suggests some additional functionality that is not present. For example, no data is stored for children (i.e., only partial family information is stored).
5	There is no provision for mass entry and editing of data by office staff. In order to accomplish this, direct access to the database is necessary.

Table 5. Limitations of the application.

5. Ongoing Development

As the organization is continuing to grow, opportunities present themselves for modifications and future work on the tool. Some specific areas in which this application will have features added or modified are the reporting and scheduling modules.

In particular, the office staff have a large body of reports that existed in the current database. While the office staff are not sufficiently proficient with Access to create their own reports, a feature has been built that allows them to create customized reports within a specific domain. This functionality does not yet exist within the project but should be developed in the near future.

Other departments having specific non-scheduling needs have shown interest in the extensibility of the application and have proposed add-on modules. For example, the youth ministry has a Teen Center open on Saturdays. Along with scheduling staffing, they would like to be able to administer the snack bar, specifically the ability to log in and track inventory, sales, and other statistics pertaining to the snack bar. There has been additional talk of similar work for a church information center and cappuccino bar.

Another area of ongoing development is the application of suitable technologies to make the interface more intuitive, consistent, and useful. Users are familiar with navigating the Internet, and are also, to some extent, familiar with the use of personal computers (it is apparent that the latter is no longer a necessary precursor of the former). However, as the understanding of usability of web-based applications grows, new technologies appear that provide increased ease-of-use and greater efficiency. AJAX and Flash are two such technologies.

AJAX is present in a limited capacity within the application in its current iteration. It allows the user to edit some fields in real-time, automatically transmitting updated data back to the database without reloading the page. However, making this functionality more widespread would make the application more consistent and more user-friendly.

Flash is not presently in use in the application, but the ability to create a user interface that is highly customized to the given application shows a great deal of promise for easing the learning curve.

At present, however, as the application is not yet fully deployed, most ongoing maintenance is of a tweaking nature. As more individuals start using the application, areas of weakness can be identified and corrected. As time moves forward, final testing will move back towards an ongoing development phase.

6. Conclusion

The application described here is the result of a great deal of research and development. While not perfect, it solves a significant problem at the sponsoring organization and provides an excellent position for ongoing development. This development will continue to improve the efficiency and efficacy of the sponsor.

The application itself, being web-based, differs in a number of ways from a traditional software product. As users are capable of creating and viewing schedules from anywhere, they are not limited by the hours and accessibility of the sponsor facilities. Furthermore, they are not limited by installation issues encountered with traditional software. As the software is web-based, updates are simple to deploy, allowing the application to be up-to-date at all times.

While a learning curve still exists, there is consensus among testers that, after a minimum of tutorial work, the interface becomes clear. Upon acclimating to the system, testers are capable of moving around quickly and effectively, able to create and modify schedules with no difficulty.

This tool, created for a single organization, can easily be adapted to any organization with a reasonably similar structure. As it is extensible, areas that are missing can be added. As it consists entirely of PHP code, areas that are lacking can be modified. For those organizations that could utilize such a tool, the open nature of the code allows a great deal of flexibility, encouraging adoption while reducing concerns over minor functionality issues.

Appendix A. Some Activity Diagrams

In the course of designing the application, some key use cases were developed, which were refined during the course of development. Included here are the most important use cases presented as UML activity diagrams.

A.1 Login

Logging in acts as the primary entry point of a user into the system. Once logged in, the user's credentials have been verified, and any permissions associated with the user have been granted.

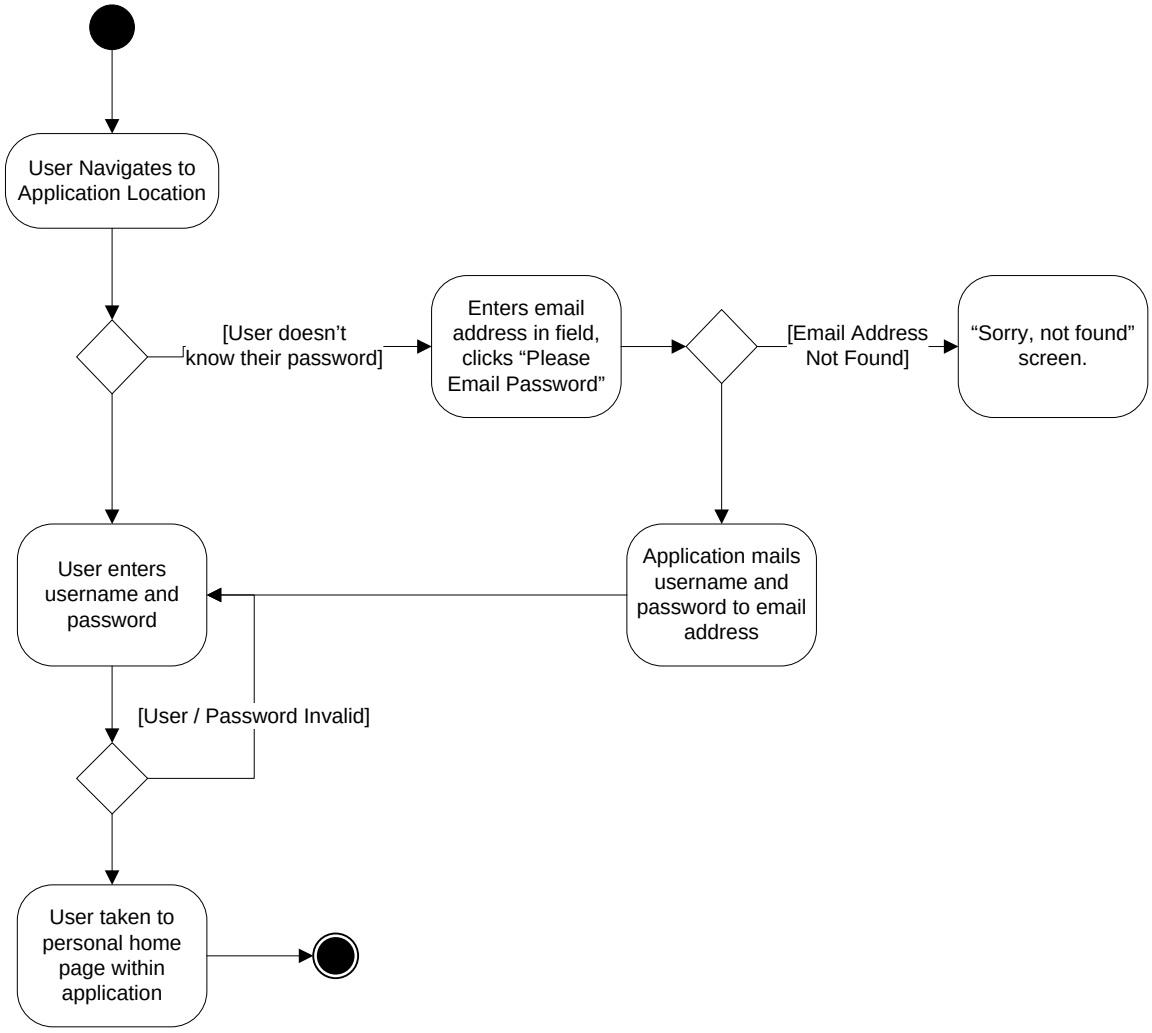


Figure 10. "Login" Activity Diagram

A.2 Set Availability

Users may specify their availability for any of the departments in which they serve. By specifying their availability, they limit how frequently they are scheduled.

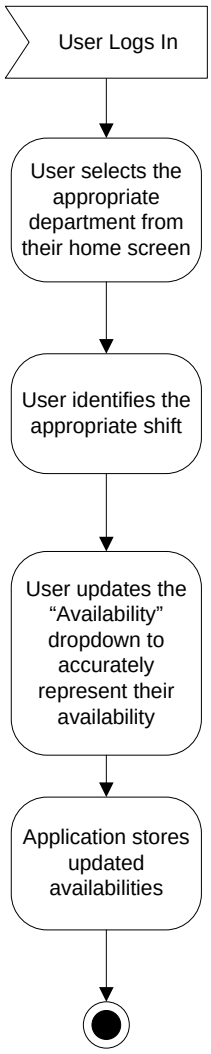


Figure 11. "Set Availability" Activity Diagram

A.3 Set Absences

Users may identify dates during which they plan on being out of town or otherwise should not be scheduled. These apply to all departments and prevent scheduling on that specific date.

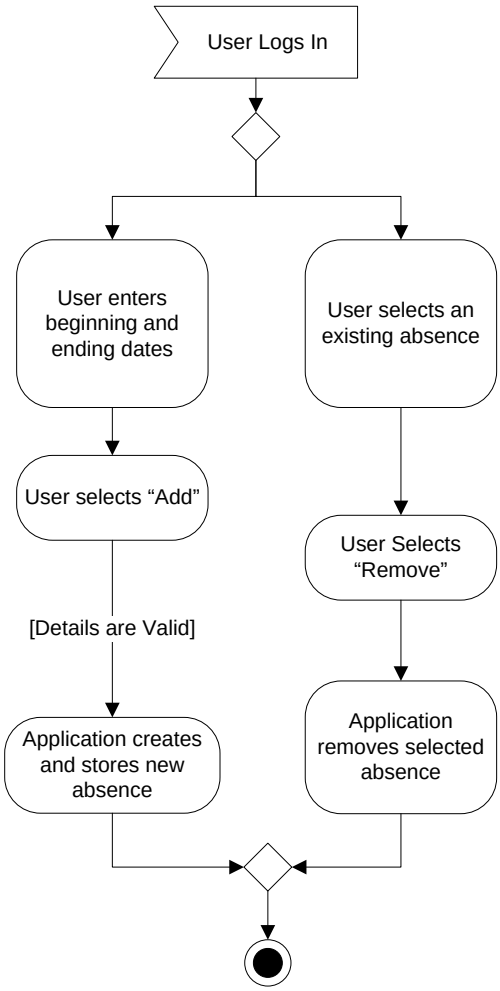


Figure 12. "Set Absences" Activity Diagram

A.4 Update Profile

Users may update their personal information, including their contact information and who has access to that information.

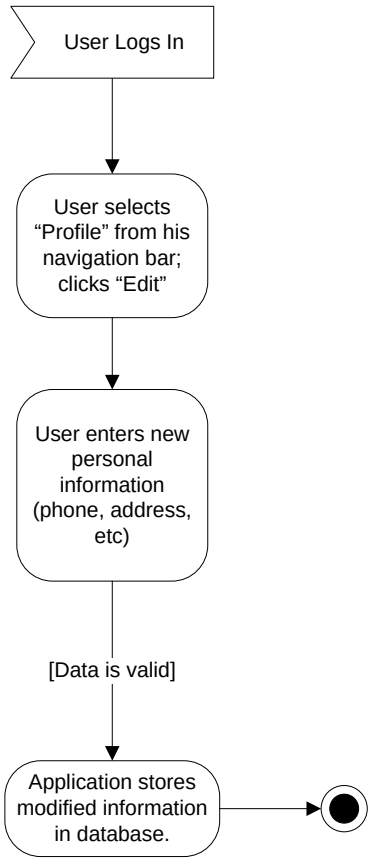


Figure 13. "Update Profile" Activity Diagram

A.5 Create or Modify Schedule

Schedule creation and modification is one of the core functionalities available within the application. It represents one of the most common use cases.

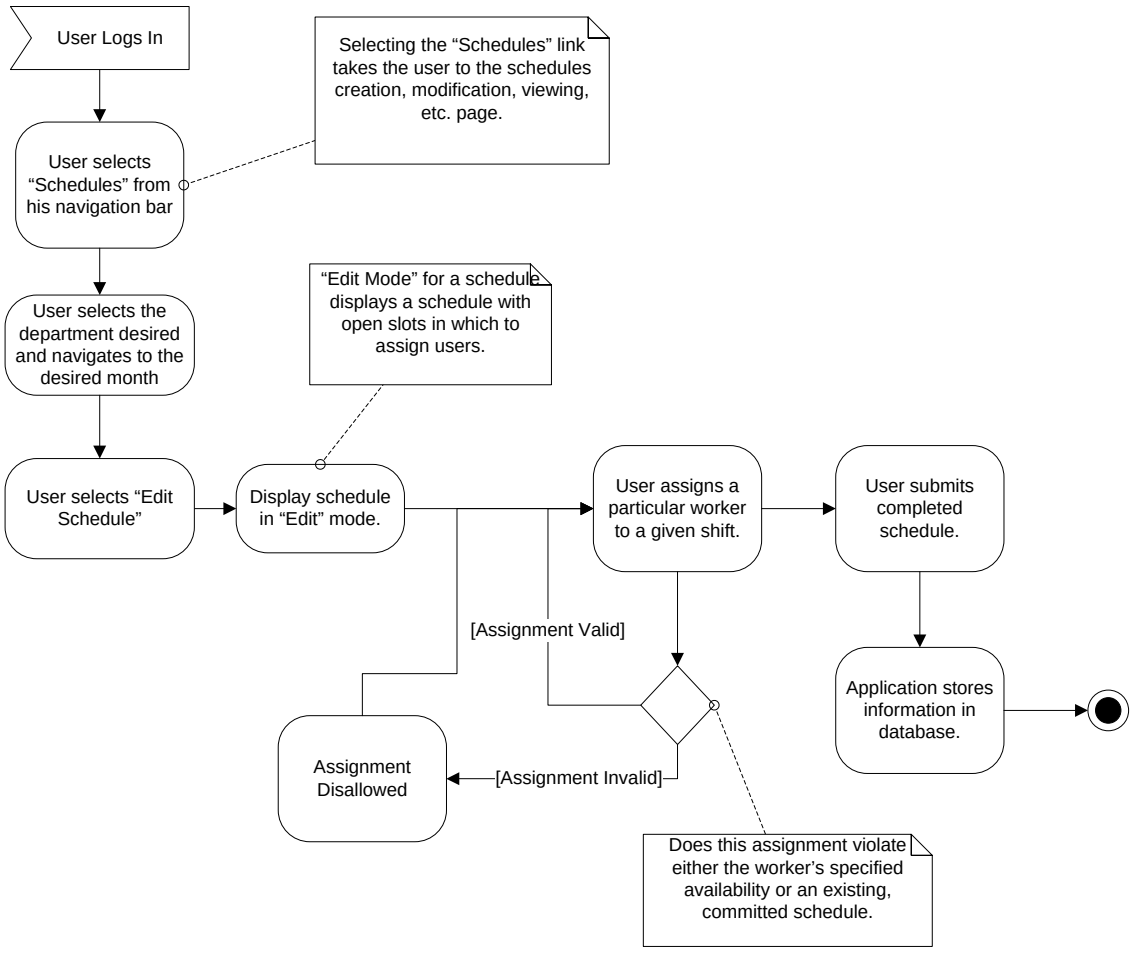


Figure 14. "Create Schedule" Activity Diagram

A.6 View a Report

Administrative users have the capacity to view a number of reports providing statistics on what departments are present, what departments individuals are active in, and so on.

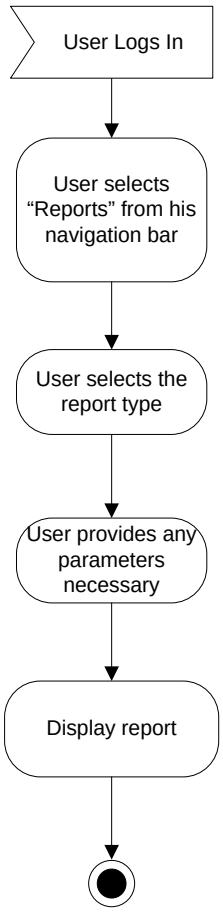


Figure 15. "View Report" Activity Diagram

A.7 View My Personal Schedule

Users have the capacity to view a schedule that is individualized for where they themselves are serving on any particular day.

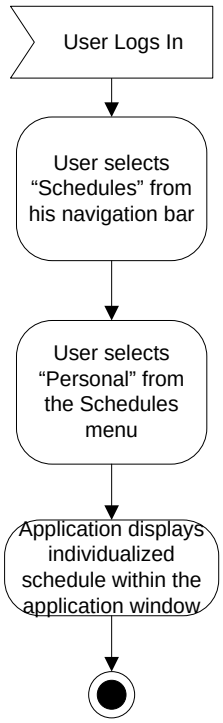


Figure 16. "View My Personal Schedule" Activity Diagram

A.8 Print My Schedule

Users have the capacity to print schedules for individual departments, all departments in which they serve, or all departments that they coordinate.

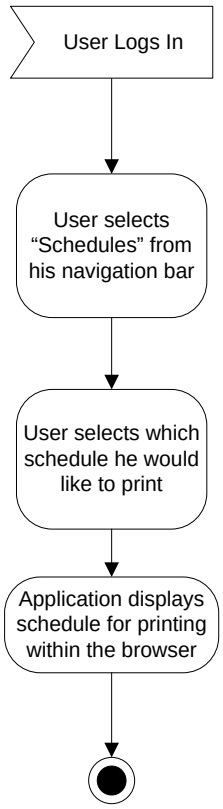


Figure 17. "Print Schedule" Activity Diagram

A.9 Add a Person or Department

Administrators on the system have the capacity to add more users or departments.

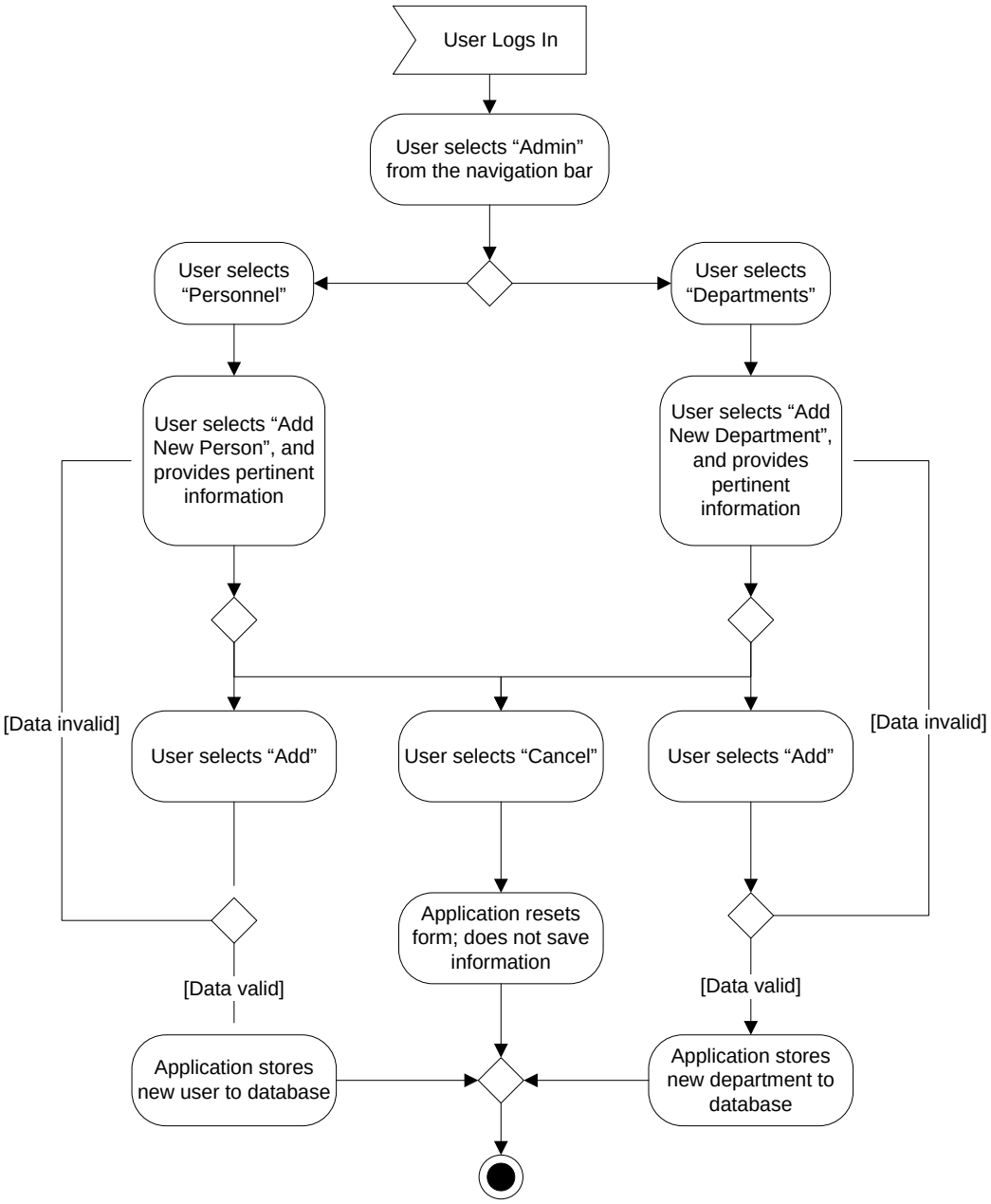


Figure 18. "Add Person or Department" Activity Diagram

A.10 Remove a Person or Department

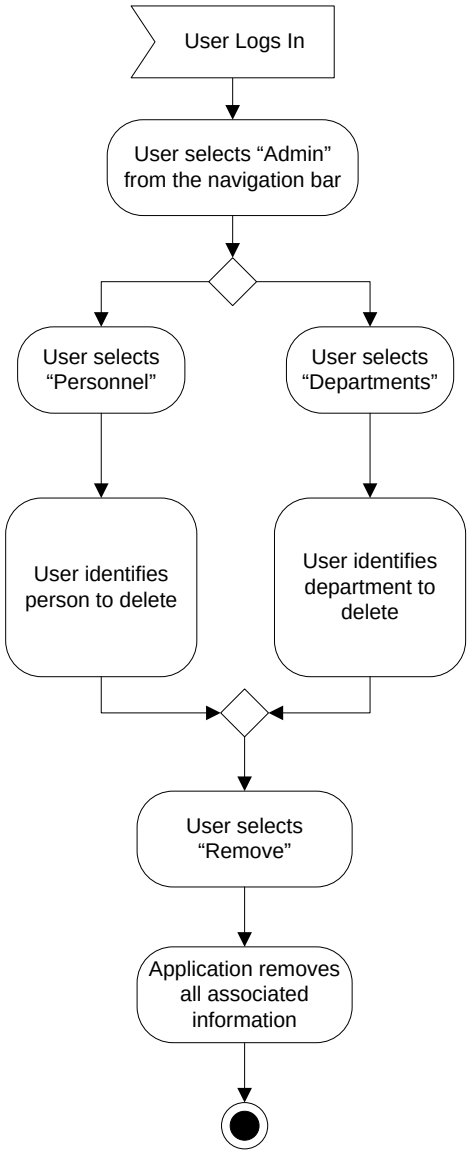


Figure 19. "Remove Person or Department" Activity Diagram

Appendix B. Interface Overview

The interface contains a number of screens. These are identified below, for reference throughout this document, as well as to provide an overview of the interface of the application.

B.1 Navigation Map

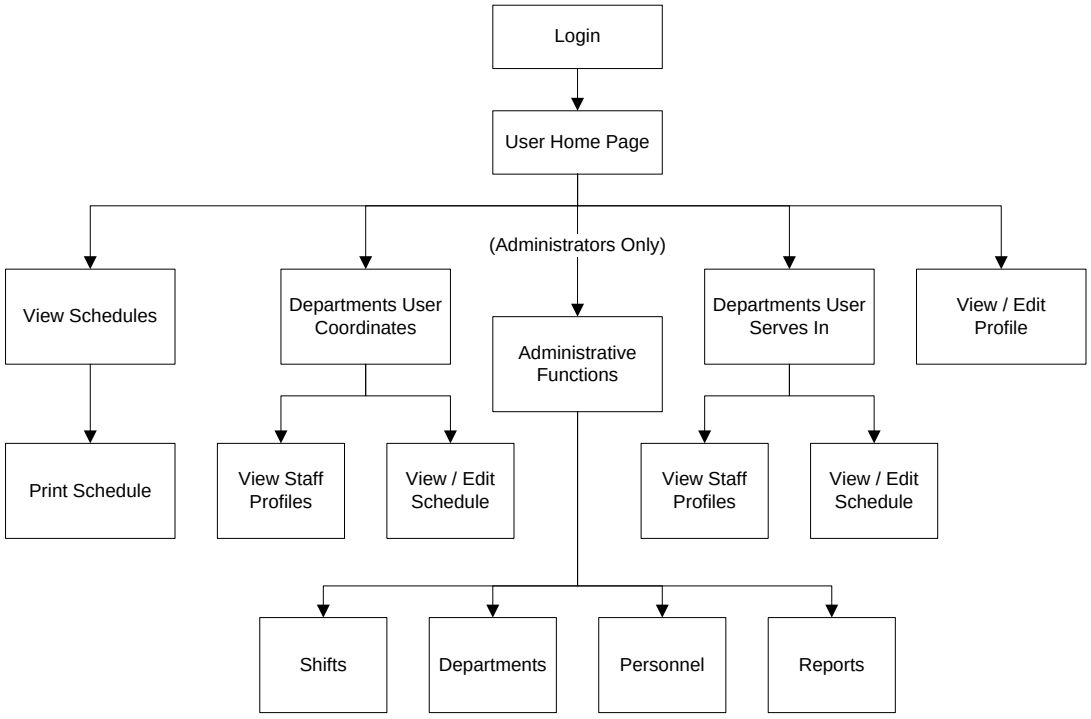


Figure 20. Navigation Map

B.2 Interface Screens

B.2.1 Login

The login screen, the first thing the user will see, allows the user to specify his or her username and password in order to gain access to the system.

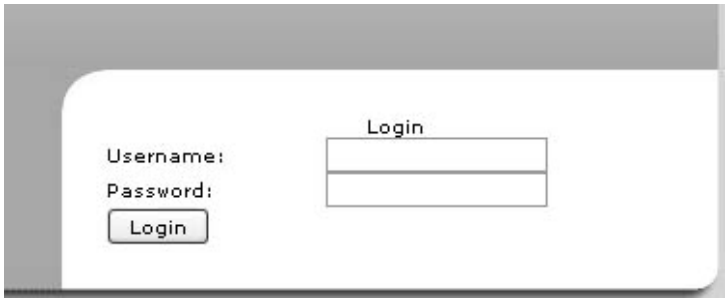


Figure 21. Login Screen

B.2.2 Home Page

The home page provides the user with quick access to all the main areas of the application.

Figure 22. User's Home Page

B.2.3 View Schedule

This screen shows the user the schedule associated with one of the departments he or she serves in. Along the left hand side, all departments that the user is involved in are shown, to allow quickly moving between departments.

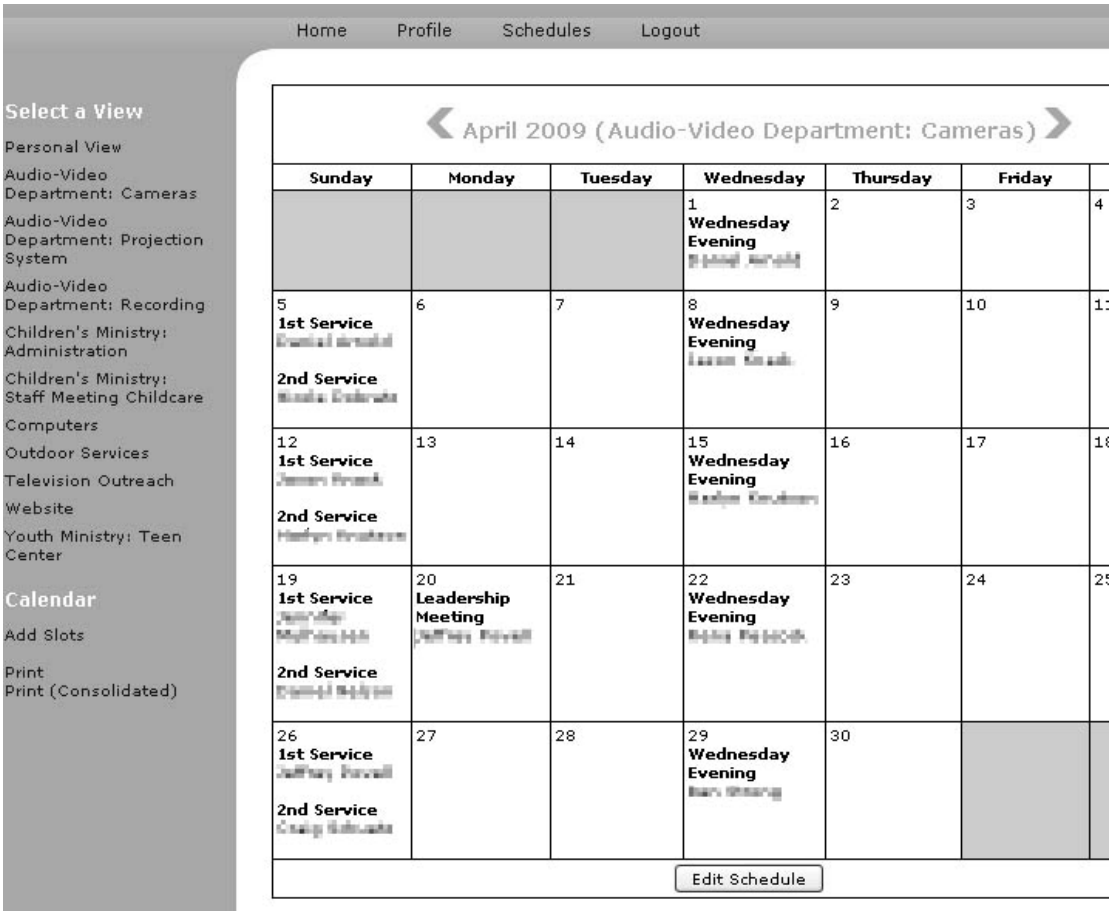
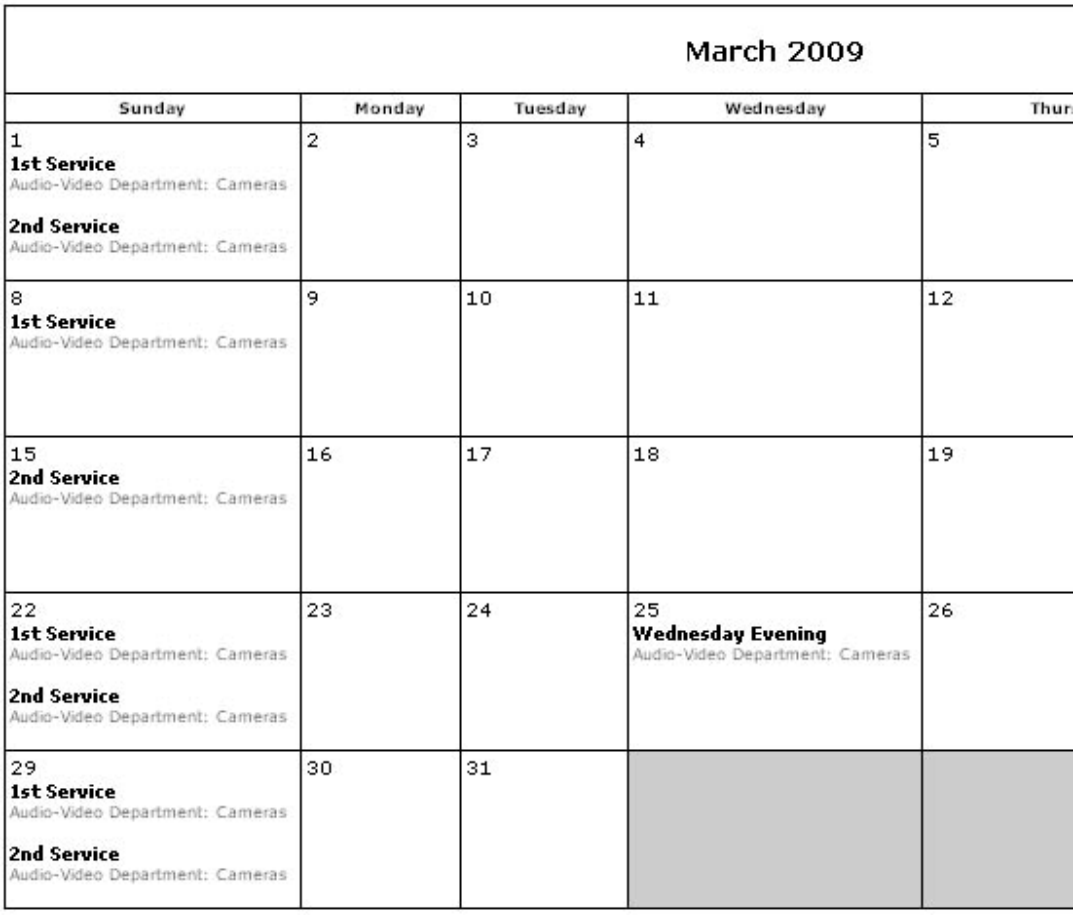


Figure 23. View Schedule

B.2.4 Print Personal Schedule

The "Personal" schedule mode is a printable schedule showing each instance that the user is scheduled and the department in which he or she is scheduled.



March 2009

Sunday	Monday	Tuesday	Wednesday	Thur
1 **1st Service** Audio-Video Department: Cameras **2nd Service** Audio-Video Department: Cameras	2	3	4	5
8 **1st Service** Audio-Video Department: Cameras	9	10	11	12
15 **2nd Service** Audio-Video Department: Cameras	16	17	18	19
22 **1st Service** Audio-Video Department: Cameras **2nd Service** Audio-Video Department: Cameras	23	24	25 **Wednesday Evening** Audio-Video Department: Cameras	26
29 **1st Service** Audio-Video Department: Cameras **2nd Service** Audio-Video Department: Cameras	30	31		

Figure 24. Print Schedule

B.2.5 Edit Schedule

Department coordinators may choose to edit a schedule, selecting individuals to serve from within the department. Upon pressing the “Submit” button, the schedule is saved.

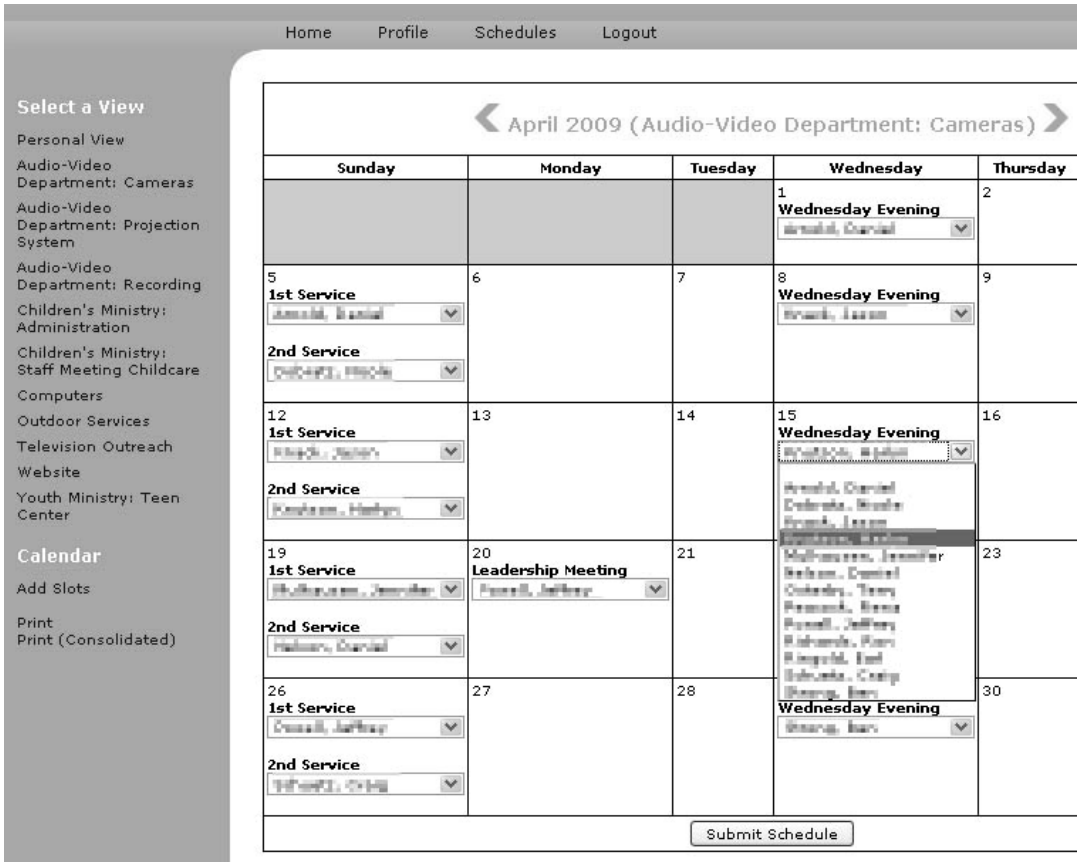


Figure 25. Edit Schedule

B.2.6 Department Coordinator Overview

When viewing a department, the coordinator of that department can select which shifts this department is active for. Coordinators may also look at the profiles of any individual within their department or go directly to viewing or editing the schedule.

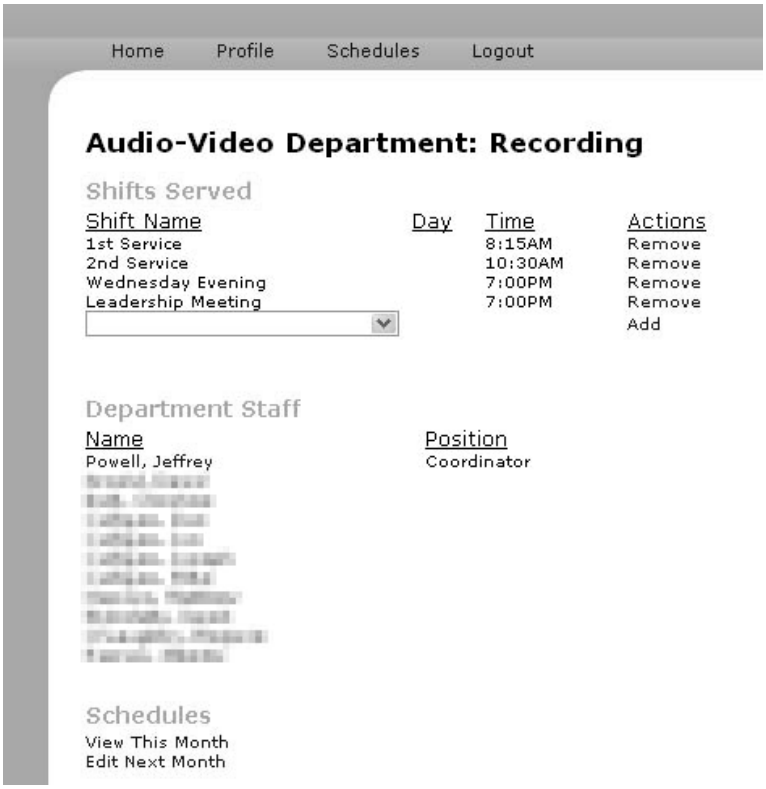


Figure 26. Department Summary (For Coordinators)

B.2.7 Department Overview

Individuals that are not department coordinators may specify their availability on a per shift basis or view the profiles of others in the department.

Figure 27. Department Summary (General)

B.2.8 Display Profile

The profile screen shows profile information for any user. On one's own profile screen, one has the option of editing the information. Additionally, users may specify how private the information is – publicly available, available to individuals within their department, available to their department coordinators only, or available only to office staff.

Figure 28. User Profile

B.2.9 Update Profile

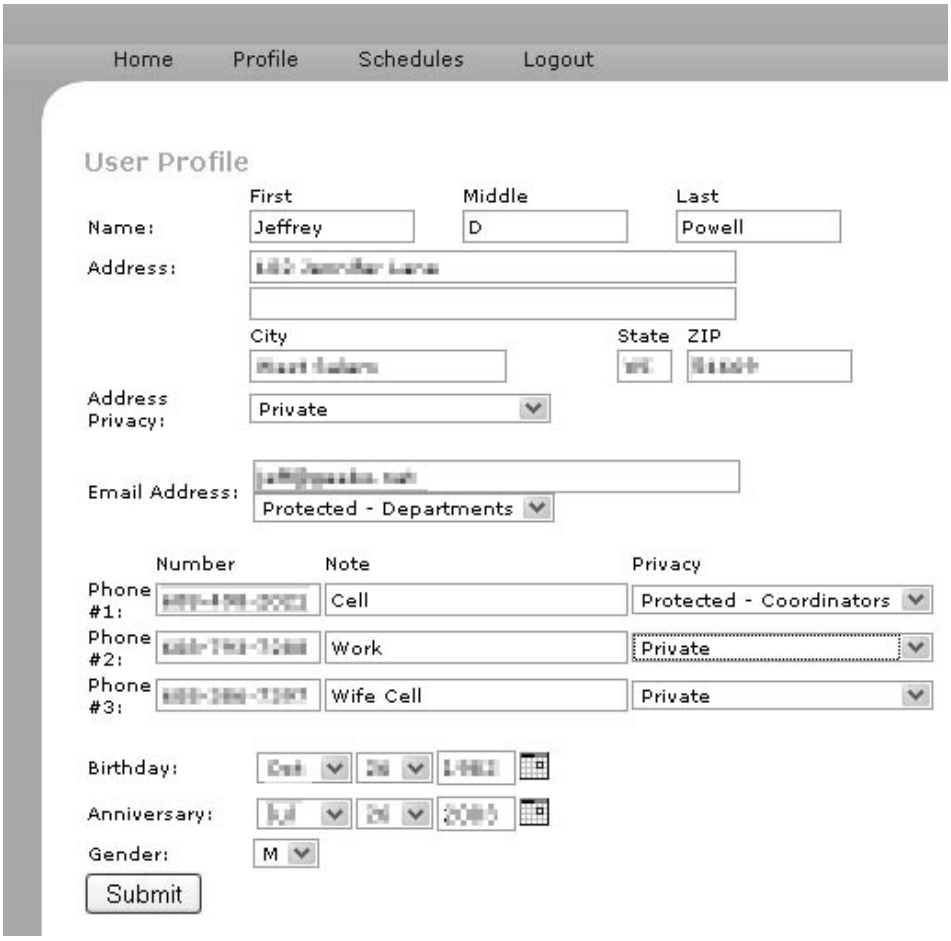


Figure 29. User Profile (Editing)

B.2.10 Change Password

The change password screen provides a mechanism whereby the user may change his or her login password.

Figure 30. Change Password

B.2.11 Administrative – Shifts

Administrative users may view all shifts that are active within the system. These shifts represent any time during which a department may be serving. From this screen, an administrator may add or remove shifts.

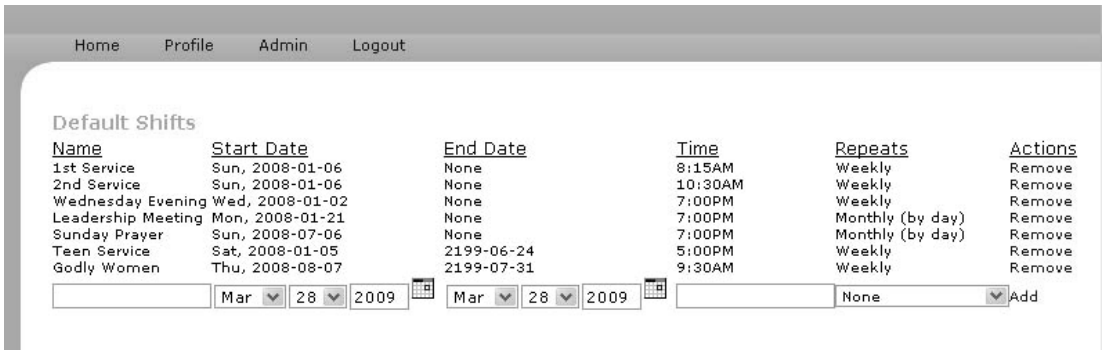


Figure 31. Administrative - Shifts

B.2.12 Administrative – Personnel

Administrative users may view all individuals present in the system. From this screen, administrators may add or remove any user.

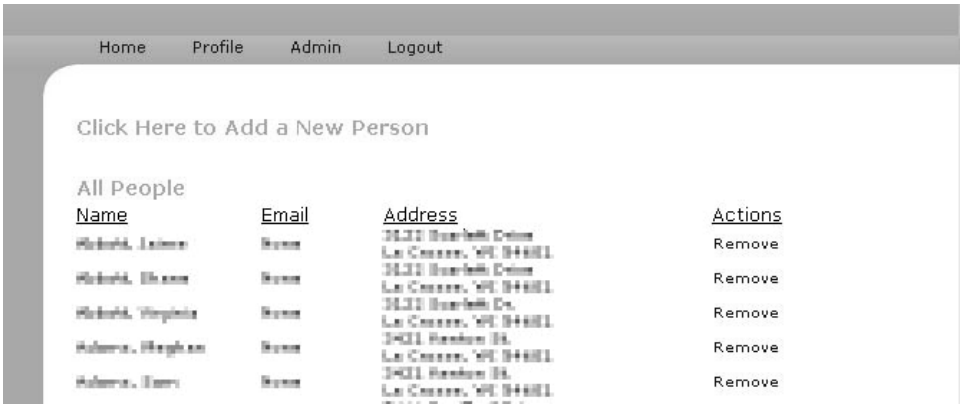


Figure 32. Administrative - Personnel

B.2.13 Administrative – Departments

Administrative users may view all departments present in the system. From this screen, administrators may add or remove any department, as well as add or remove staff from any department.

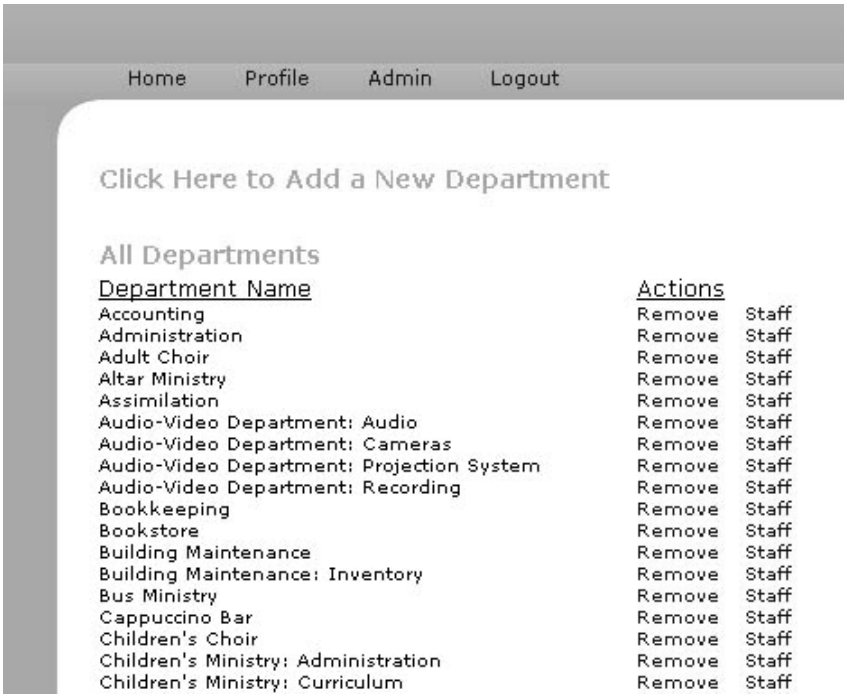


Figure 33. Administrative - Departments

B.2.14 Administrative – Reports

Administrative users have a number of reports available to them. From the reports screen, administrators may run these reports.

Figure 34. Administrative – Reports

Appendix C. Database Table Details

The following tables represent a more in-depth view of the data stored in each field shown in the ERD in Figure 6.

C.1 Table 'people'

Column Name	Description
Id	The ID of the user, and key to the table
Username	The username used to login
Password	A hash of the password, checked upon login
first_name	Given name
middle_name	Middle name
last_name	Surname
Email	Preferred email address
email_privacy	The level of privacy desired for the email address
Street	Street address
street2	Secondary street address (i.e., apartment information, c/o, etc.)
City	City
State	State
Zip	ZIP-code
address_privacy	The level of privacy desired for the street address
phone1_number	A primary phone number
phone1_note	A description indicating the nature of the number
phone1_privacy	The privacy level desired for the phone number
phone2_number	A 2nd phone number
phone2_note	An associated description
phone2_privacy	An associated privacy level
phone3_number	A 3rd phone number
phone3_note	An associated description
phone3_privacy	An associated privacy level
adminFlag	Flag indicating whether this user is an administrator within this system
Title	The standard title (Mr., Mrs., Dr., etc.)
Birthday	Birthday
Gender	Gender
Anniversary	Anniversary
spouse_id	The ID number of the spouse, if applicable
notes	Any notes regarding this individual. This field exists as a precursor to extended functionality for

	office personnel.
bg_check_date	Date a background check was completed
membership_status	Whether this individual has completed any membership classes
first_attended	When this individual first attended
date_received	When this individual became a member
code_of_conduct_signed	When this individual turned in a signed code of conduct form (required for serving in any of the various departments)
head_of_household	Whether this individual is the head of his or her household (used in office reports)
courtesy_title	Titles of courtesy, used on reports
is_deacon	Whether this individual is a congregational deacon
is_elder	Whether this individual is a congregational elder
office_staff	Whether this individual is an office staff member

Table 6. Description of database table "people".

C.2 Table ‘departments’

Column Name	**Description**
Id	The ID of the department
department_name	The name of the department
Description	Description of the department
renewal_month	Annual renewal month of the department

Table 7. Description of database table "departments".

C.3 Table ‘people_in_departments’

Column Name	**Description**
department_id	The ID of the department in which the associated individual serves
person_id	The ID of the person in the position
position_name	The name of the position (e.g., “Asst. Teacher”)
Application	Date the application was received
renewal_date	Date the most recent renewal form was received
approved_through	Date through which the renewal is valid
training_complete	Date when any required training was completed
release_from_service	Date when a release from service form was provided
adminRole	Whether this user and position constitutes an administrative position for this department

Table 8. Description of database table "people_in_departments".

C.4 Table ‘shifts’

Column Name	Description
Id	The ID of the shift
Time	The time that the shift starts
Duration	The duration of the shift
Name	The name of the shift
repeat_type	How often the shift occurs
start_date	The effective date of the shift
end_date	When to stop scheduling this shift
assoc_dept	If this shift was created specifically for a single department, the department ID of that department (prevents use of this shift by other departments)

Table 9. Description of database table "shifts".

C.5 Table ‘department_shifts’

Column Name	Description
department_id	The department ID associated with this shift
shift_id	The shift ID associated with the department

Table 10. Description of database table "department_shifts".

C.6 Table ‘absences’

Column Name	Description
Id	The ID number of the absence
person_id	The ID of the person to be absent
start_date	The start date of the absence (inclusive)
end_date	The ending date of the absence (inclusive)
Note	Any note related to the absence

Table 11. Description of database table "absence".

C.7 Table ‘availability’

Column Name	Description
person_id	ID of the person
department_id	ID of the department
shift_id	ID of the shift
availability_id	ID of the availability type

Table 12. Description of database table "availability".

C.8 Table ‘availability_classes’

Column Name	Description
Id	The ID of the availability class
availability_name	Name of the availability class
Description	A concise description of the availability class
max_sched	The maximum number of times a person using this availability class can be scheduled for a given shift during a given scheduling period

Table 13. Description of database table "availability_classes".

C.9 Table ‘schedules’

Column Name	Description
person_id	ID of the scheduled individual
department_id	ID of the department
shift_id	ID of the shift
Month	Month of the schedule
Date	Day within the month
Year	Year of the schedule

Table 14. Description of database table "schedules".