Go Navy!

Architectural Documentation
Version 1.0

Project Owner:
Erik Shepard

Developers:
Gustavo Hammerschmidt
Jonathan Scott
Erik Shepard

CS 441 Software Engineering
Section 01
California State University, San Marcos
6th March 2020

Contents

1 Introduction

The San Diego Naval Reserve Officer Training Corps (NROTC) is one of 63 consortiums in the country which produces the majority of military officers in the United States Navy and Marine Corps. The composition of the unit consists of an active duty permanent staff, a student staff, and a student body. This large and very diverse program manages over 250 personnel in training and military development.

Many different training events occur frequently over every academic term which are requirements for all students and the statistics from each event must be well documented. Student schedules are also documented to ensure minimal conflicts occur for accountability when a student cannot attend an event. A routing process is in place for a student to request in advance for approval to miss an event should the occasion arise.

Currently, all managing staff utilize stand-alone documents and programs such as Microsoft Excel Spreadsheets and Word documents, Google Sheets, Adobe Acrobat Forms, Blackboard, and email to maintain the large amount of data throughout every term. Errors occur frequently when documenting attendance rosters, sending electronic correspondence, and inconsistent formatting due to misinformation, typos, and incorrect personnel data among other reasons. Much of the documentation that occurs contains consistent information but is reproduced many times, increasing the odds for an error to occur.

A custom website application is desired to be able to encapsulate a central point of data for proper documentation. The application must be accessible and user-friendly for all personnel via internet. It must also be secure to protect sensitive personally identifiable information (PII).

To solve this problem, we have decided to make a software based on the client-server architecture. There is a user: which, in the client-server model, would be the client. The user is responsible for taking care and managing the software, as well as it can give orders to other users and manage its problems through the application, some functions will be available to some users and others will not, this will be defined according to the user rank: that will dictates if the user is allowed or not to make such operation. Users use the application to solve its problems, check its information, etc(for a more detailed version of the users capabilities, check the section two of the requirements document). The server is the part of the application that will be running on server-dedicated computer, it will provide interfaces for its users and access the database to manage data. The database is the part of the software that will save all the application data.

The high-level architecture for the system is shown in Figure 1-1.

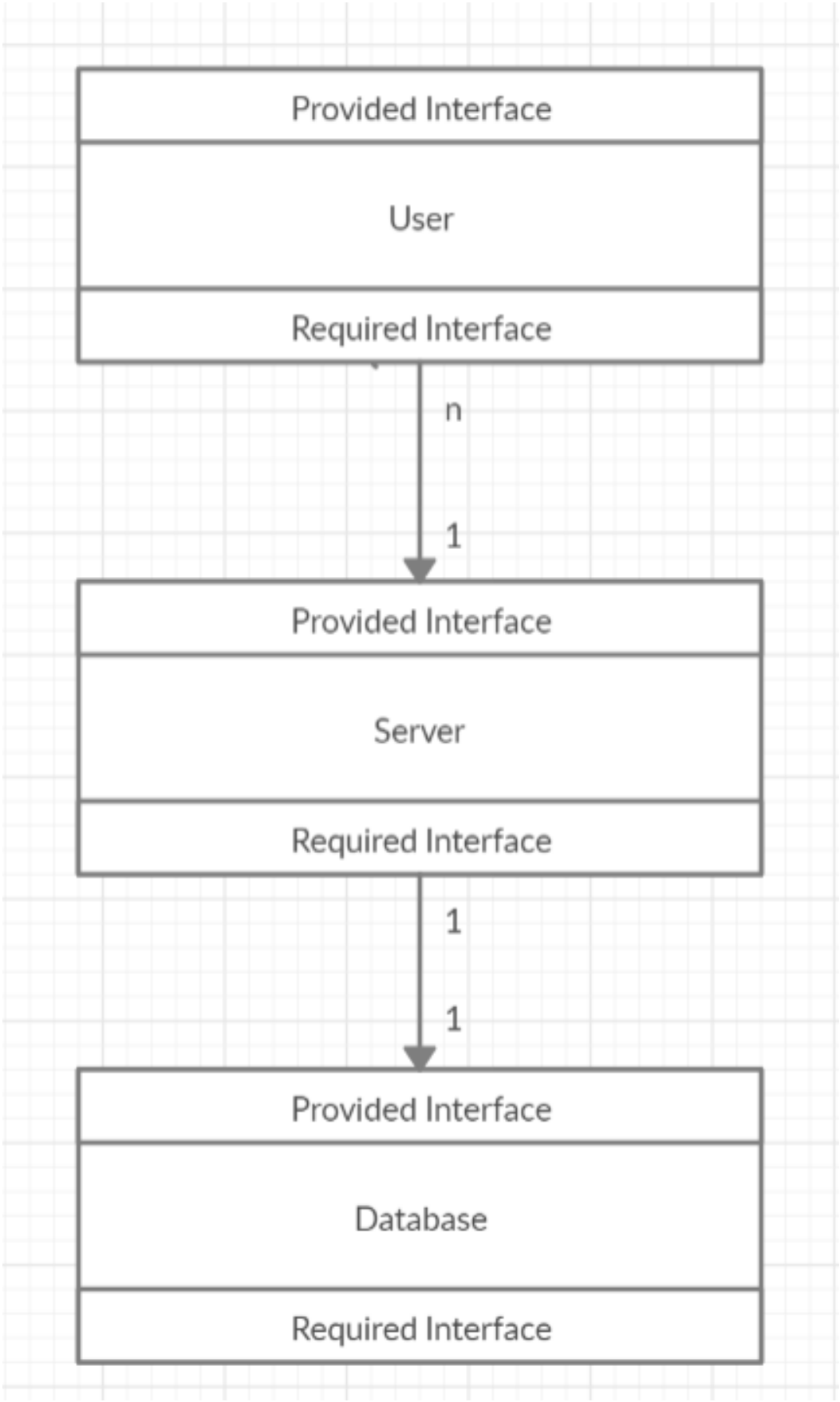

Figure 1.1: Introduction: High Level Architecture

1.1 Server

Please see Chapter 2 for a detailed description of this module.

1.2 Database

Please see Chapter 3 for a detailed description of this module.

1.3 User

Please see Chapter 4 for a detailed description of this module.

2 Server Design

2.1 Server Module

Purpose

The purpose of this module is to handle all the information coming from the user module, as well as manipulate them, store and access them in a database module.

Rationale

This module is created to centralize and encapsulate all data storage management and retrieval duties on the system: respond to the users command. It includes a communication module and a user module(for more information on these sub-modules, please check the following sub-sections).

High-level Server Design

The server module is broken down into lower-level modules, as shown in Figure 2-1. All the information coming from the user module is handled by the communication modules. The User module handles, respectively, user functions. Each of these modules will be described in detail in following sections.

Required Interface

This module's required interface is the provided interface of the Database Module.

Provided Interface

This module provides one interface: the user interface, that is to be used by its respectively module: the user module (this will be defined in chapters 4).

For more detail, please see these modules' descriptions.

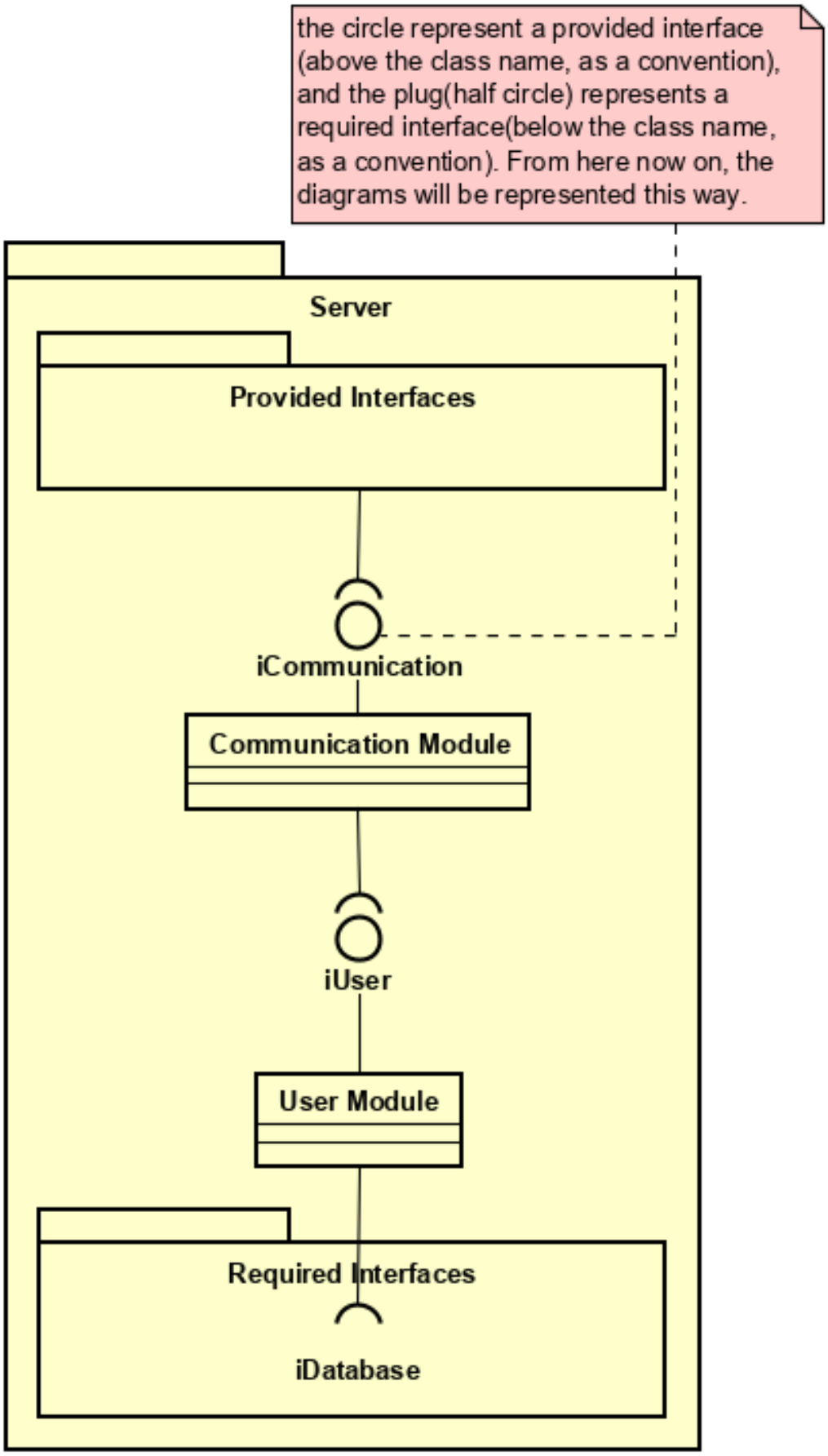

Figure 2.1: Chapter 2: Server Module

2.2 Communication Module

Purpose

The purpose of the communications module is to provide a communication link between the users and the server. It represents the part of the communications link that resides on the server side of the system.

Rationale

This module is created to centralize and encapsulate network communication duties between clients and the server.

Required Interface

This module's required interface is the provided interface from the user module.

Provided Interface

This module provides an interface to the following :

- User Module.

2.3 User Module

Purpose

The purpose of these module is to authenticate users, and to enable them to call the server functions to manage data.

Rationale

This module is created to centralizes functions related to the user module; in a way, it can interface and communicate with the server module.

High-level User Module design

The user module is broken down into high-level modules, as shown in Figure 2.2

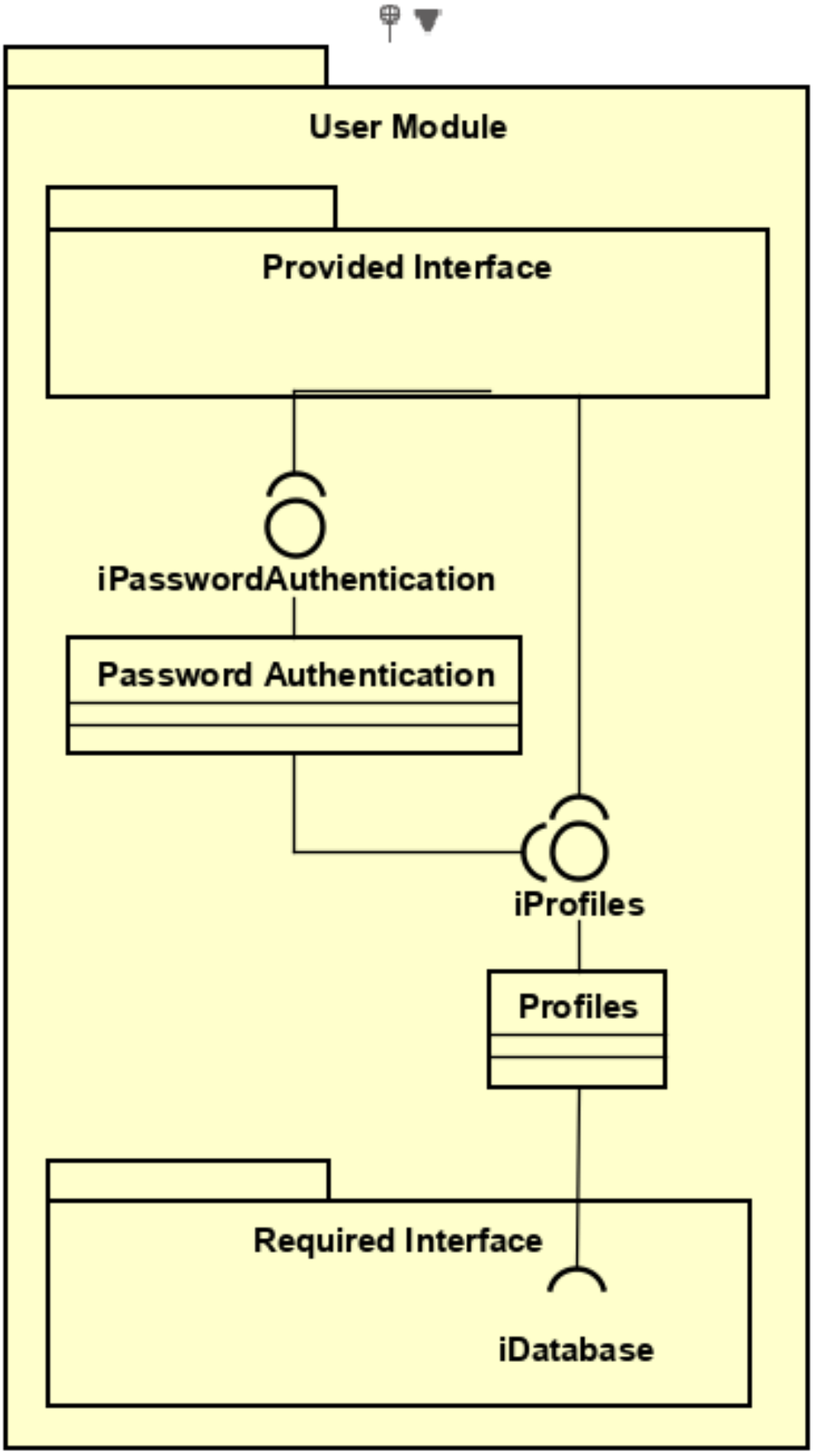

Figure 2.2: User Module High-level Architecture

Required Interface

This module requires the following modules:

- Database module.

Provided Interface

This module provides the following sub-modules:

- Password Authentication.
- Profiles.

2.3.1 Password Authentication Module

Purpose

The purpose of this module is to provide an authentication service, allowing callers to determine whether a username/password combination is valid, and a change-password service, allowing users to change their passwords.

Rationale

This module is created to centralize and encapsulate password management and authentication services.

Required Interfaces

This module requires the following modules:

- Database module.

```
boolean userExists(String userID);
```

Provided Interfaces

```
void setPassword(String userId, String password)
throws NoSuchUserException;
```

Description: Stores a user password of the given user ID.

Parameters:

- userID: User ID of the user whose password is being stored.
- password: The password for that user.

Exceptions:

- NoSuchUserException: If the given user does not exist.

```
boolean logon(String userID, String password)
throws BannedUserException;
```

Description: Determines whether a given user/password combination is valid or not. For security reasons, the incorrect portion of the combination (i.e. user or password)

is not given.
Parameters:

- userID: User ID of the user whose password is being checked.
- password: The presumed password for that user.

Returns:

- True if the user ID/password combination is correct, false otherwise.

Exceptions:

- RemovedUserException: If the given user's access has been removed.

```
void logoff(String userID) throws NoSuchUserException,
UserNotLoggedOnException;
```

Description: Logs a user off the system.
Parameters:

- userID: User ID of the user who is logging off.

Exceptions:

- NoSuchUserException: If the given user does not exist.
- UserNotLoggedOnException: If the given user is not logged on.

2.4 Profiles Module

Purpose

The purpose of this module is to provide access to and a persistent data store for all user profiles in the system.

Rationale

This module is created to centralize and encapsulate user profiles.

Required Interfaces

This module requires the following modules:

- Database module.

Provided Interfaces

This module provides the following sub-modules:

- Chits.
- Scheduling.
- Training.

2.4.1 Chit Module

Purpose

The purpose of this module is to provide access to and a persistent data store for all chits in the system.

Rationale

This module is created to centralize and encapsulate chits.

Required Interfaces

This module requires the following modules:

- Database module.

Provided Interfaces

```
void createChit(String userId, date beginDate, date endDate, int
beginTime, int endTime, String reason, String justification, boolean
travel, File doc)
```
Description: Stores a user password of the given user ID.

Parameters:

- userID: User ID of the user whose password is being stored.
- password: The password for that user.

Exceptions:

- NoSuchUserException: If the given user does not exist.

2.4.2 Scheduling Module

Purpose

The purpose of this module is to provide access to and a persistent data store for scheduling data in the system.

Rationale

This module is created to centralize and encapsulate scheduling data.

Required Interfaces

This module requires the following modules:

- Database module.

Provided Interfaces

```
void createDuty(String dutyName, String userID) throws UserNotAuthorized,
UserNotLoggedOnException;
```

Description: Creates a scheduling event.
Parameters:

- dutyName: Name of the type of duty to schedule
- userID: Name of the user that is scheduled for the duty.

Exceptions:

- UserNotAuthorized: If the user is not high enough rank to create an event.
- UserNotLoggedOnException: If the given user is not logged on.

2.4.3 Training Module

Purpose

The purpose of this module is to provide access to and a persistent data store for all training events in the system.

Rationale

This module is created to centralize and encapsulate training events.

Required Interfaces

This module requires the following modules:

- Database module.

Provided Interfaces

```
Event createEvent(DateTime begin, DateTime end, String title,
String description) throws UserNotAuthorized,
UserNotLoggedOnException;
```

Description: Creates a training event.
Parameters:

- begin: Date and time the event begins.
- end: Date and time the event end.
- title: Title of the event.
- description: Description of the event.

Exceptions:

- UserNotAuthorized: If the user is not high enough rank to create an event.

- UserNotLoggedOnException: If the given user is not logged on.

```
List<Event> listEventsRange(Date begin, Date end)
throws UserNotLoggedOnException
```
Description: Returns a list of events in the specified range.
Parameters:

- begin: Date to search after.
- end: Date to search before.

Exceptions:

- UserNotLoggedOnException: If the given user is not logged on.

```
void updateEvent(Event event) throws UserNotAuthorized,
UserNotLoggedOnException;
```

Description: Updates a training event.
Parameters:

- event: The event to update

Exceptions:

- UserNotAuthorized: If the user is not high enough rank to create an event.
- UserNotLoggedOnException: If the given user is not logged on.

```
void deleteEvent(Event event) throws UserNotAuthorized,
UserNotLoggedOnException;
```

Description: Deletes a training event.
Parameters:

- event: The event to delete

Exceptions:

- UserNotAuthorized: If the user is not high enough rank to create an event.
- UserNotLoggedOnException: If the given user is not logged on.

---

3 Database

3.1 Communication Module

Purpose

The purpose of the communications module is to provide a communication link between the server and the database. It represents the part of the communications link that resides on the database side of the system.

Rationale

This module is created to centralize and encapsulate user data, and to make it easier for the server to store, manage and return data to the user.

Required Interface

This module has no required interface.

Provided Interface

This module provides an interface to the following :

- Server Module.

3.2 Database Module

Purpose

The purpose of the database modules is to load and store data to and from XML files.

Rationale

This modules is created to store and retrieve data.

High-level Database design

The high-level design of the Database module is shown on the Figure 3.1.

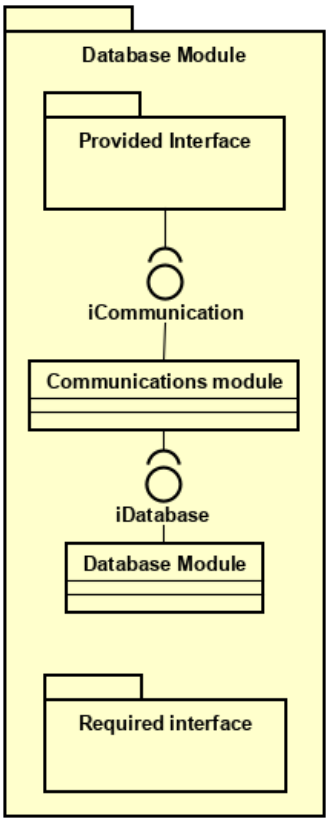

Figure 3.1: High-level Database Module Design

Required Interface

```
void setCreateChit(String userId, date beginDate, int beginTime,
date endDate, int endTime, String reason, String justification,
boolean travel, boolean doc) throws MissingInfoException;
```

Description: Stores data for a new chit.

Parameters:

- userID: User ID of the user who is creating the chit.
- beginDate: The beginning date for the period of the requested chit.
- beginTime: The beginning time for the period of the requested chit.
- endDate: The end date for the period of the requested chit.
- endTime: The end time for the period of the requested chit.
- reason: Defines the reason type for the chit that is being requested.
- justification: Comments by the requester for clarifying details.
- travel: Identifies if travel is being requested.
- doc: Identifies if a document will be saved.

Exceptions:

- MissingInfoException: If the given user does not provide required information.

```
void setAttachChitDocument(String chitID, File document) throws
SizeLimitException;
```

Description: Attaches a file to a chit.

Parameters:

- chitID: Chit ID for the chit that the document will be attached to.
- document: The file that will be attached.

Exceptions:

- SizeLimitException: If the file size exceeds 100MB.

```
void setTravelDetails(String chitID, int transportationMode, int
distance, String primaryPhone, String primaryAddress, String altPhone,
String altAddress) throws MissingInfoException;
```

Description: Retrieves data for a chit by the originator's user ID.

Parameters:

- chitID: Chit ID for the chit that the document will be attached to.
- transportationMode: Integer option for the type of transportation.
- distance: Number of miles to destination.
- primaryPhone: Primary phone number.
- primaryAddress: Primary address of destination.
- altPhone: Alternate phone number.
- altAddress: Alternate address of destination.

Exceptions:

- MissingInfoException: If the chit ID is invalid.

```
Chit setNextRoutingUser(String nextUserID)
```

Description: Sets the user of the next user in the router process.

Parameters:

- nextUserID: User ID of the next user.

`Chit setChitRoutingComment(Date date, Time time, String currentUser, String currentUserComments, String recommendation, String gotoUserID)`

Description: Sets the comment and recommendation of a chit by the chain of command.

Parameters:

- date: Date that the comment is made.
- time: Time that the comment is made.
- currentUser: User ID of the user making the comment.
- currentUserComments: Comment from the current user.
- recommendation: The current user's recommendation.
- gotoUserID: The user ID for the next person who needs to make a recommendation.

`Chit setChitStatus(String status)`

Description: Sets the status of the chit.

Parameters:

- status: Defines whether the chit status is complete, rework, or routing.

Provided Interface

This module provides the following interfaces to the Server Module:

`boolean getApprovalAuthority(String userID)`

Description: Checks if the user has approval authority for chits. **Parameters**:

- userID: The user ID to check if they have permissions to approve a chit.

`Chit getChitByID(String chitID) throws NoIDException;`

Description: Retrieves data for a chit by the chit ID.

Parameters:

- chitID: Chit ID that is being retrieved.

Exceptions:

- NoIDException: If the chit ID is invalid.

```
List<Chit> getChitByUserID(String userID) throws
NoChitsByUserIDException;
```

Description: Retrieves data for a chit by the originator's user ID.

Parameters:

- userID: User ID of the originator for all chits that are being retrieved.

Exceptions:

- NoChitsByUserIDException: If there are chits to retreive with the user's ID.

```
List<Chit> getChitByAwaitingAuthority(String userID) throws
NoChitsByAAException;
```

Description: Retrieves all chits that are awaiting approval by the user's ID.

Parameters:

- userID: User ID of the user who is listed as the awaiting approval for any chits.

Exceptions:

- NoChitsByAAException: If there are no chits that are awaiting approval by the user's ID.

```
void getNextSupervisor(UserID userID) throws NoSupervisorException;
```

Description: Retrieves the current user's supervisor.

Parameters:

- userID: User ID of the originator for all chits that are being retrieved.

Exceptions:

- NoSupervisorException: If the user has no supervisor.

---

4 User

4.1 User Module

4.1.1 Purpose

The purpose of this module is to provide the user interface and view functions for the system. This is the software with which the user directly interacts. It communicates with the server to retrieve and modify persistent data when necessary.

4.1.2 Rationale

This module is created to display data from the server module. This kind of data includes users, chits, events, etc.

4.1.3 High-Level Module Design

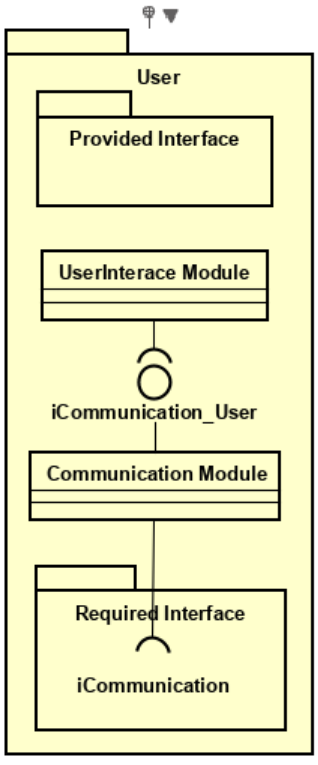

Figure 4.1: Chapter 4: User Module

4.1.4 Required Interface

The required interface of this module is the same as the required interface of the Communications module (see section 2.2)

4.1.5 Provided Interface

This module has no provided interfaces.

4.2 UserInterface Module

4.2.1 Purpose

The purpose of this module is to provide a User Interface that staff and students can interact with.

4.2.2 Rationale

This module is created to display data that is received from the communication module.

4.2.3 High-Level Module Design

The User module is broken down into lower-level modules as shown **

4.2.4 Required Interface

The required interface of this module is the same as the provided interface of the Communications module.

4.2.5 Provided Interface

This module has no provided interface.

4.3 Communications Module

4.3.1 Purpose

The purpose of this module is to provide communication services between the user client of the system and the server. This module represents the part of the communications link that resides on the user client.

4.3.2 Rationale

This module is created to centralize and encapsulate network communication duties between the user client and the server.

4.3.3 Required Interface

This module's required interface is the same as its provided interface, but it makes calls to the server's communication module via an implementation-dependent network protocol (HTTP).

```
void setPassword(String userId, String password)
throws NoSuchUserException;
```

Description: Stores a user password of the given user ID.

Parameters:

- userID: User ID of the user whose password is being stored.
- password: The password for that user.

Exceptions:

- NoSuchUserException: If the given user does not exist.

```
boolean logon(String userID, String password)
throws BannedUserException;
```

Description: Determines whether a given user/password combination is valid or not. For security reasons, the incorrect portion of the combination (i.e. user or password) is not given.
Parameters:

- userID: User ID of the user whose password is being checked.
- password: The presumed password for that user.

Returns:

- True if the user ID/password combination is correct, false otherwise.

Exceptions:

- RemovedUserException: If the given user's access has been removed.

```
void logoff(String userID) throws NoSuchUserException,
UserNotLoggedOnException;
```

Description: Logs a user off the system.
Parameters:

- userID: User ID of the user who is logging off.

Exceptions:

- NoSuchUserException: If the given user does not exist.
- UserNotLoggedOnException: If the given user is not logged on.

```
Event createEvent(DateTime begin, DateTime end, String title,
String description) throws UserNotAuthorized,
UserNotLoggedOnException;
```

Description: Creates a scheduling event.
Parameters:

- begin: Date and time the event begins.
- end: Date and time the event end.
- title: Title of the event.
- description: Description of the event.

Exceptions:

- UserNotAuthorized: If the user is not high enough rank to create an event.
- UserNotLoggedOnException: If the given user is not logged on.

```
List<Event> listEventsRange(Date begin, Date end)
throws UserNotLoggedOnException,
```

Description: Returns a list of events in the specified range.
Parameters:

- begin: Date to search after.
- end: Date to search before.

Exceptions:

- UserNotLoggedOnException: If the given user is not logged on.

```
void updateEvent(Event event) throws UserNotAuthorized,
UserNotLoggedOnException;
```

Description: Creates a scheduling event.
Parameters:

- event: The event to update

Exceptions:

- UserNotAuthorized: If the user is not high enough rank to create an event.
- UserNotLoggedOnException: If the given user is not logged on.

```
void deleteEvent(Event event) throws UserNotAuthorized,
UserNotLoggedOnException;
```

Description: Creates a scheduling event.
Parameters:

- event: The event to delete

Exceptions:

- UserNotAuthorized: If the user is not high enough rank to create an event.
- UserNotLoggedOnException: If the given user is not logged on.

```
void createChit(String userId, date beginDate, date endDate, int
beginTime, int endTime, String reason, String justification, boolean
travel, File doc)
```

Description: Stores a user password of the given user ID.

Parameters:

- userID: User ID of the user whose password is being stored.
- password: The password for that user.

Exceptions:

- NoSuchUserException: If the given user does not exist.

```
void createDuty(String dutyName, String userID) throws UserNotAuthorized,
UserNotLoggedOnException;
```

Description: Creates a scheduling event.
Parameters:

- dutyName: Name of the type of duty to schedule
- userID: Name of the user that is scheduled for the duty.

Exceptions:

- UserNotAuthorized: If the user is not high enough rank to create an event.
- UserNotLoggedOnException: If the given user is not logged on.

4.3.4 Provided Interface

The provided interface of this module is the same as the required interface of the UI module.