**Dental Clinic Management System**

By: Mazen Abid, Ethan Goski, Froillo Raquinio

**1 Introduction**

The Campus Dental Clinic at the University of Regina faces many problems including managing patient records, making appointments, billing and maintaining inventory. As a result, this poses a burden on administration leading to longer waiting times for patients as well as mistakes in record keeping. In response to this problem, we developed a full-fledged software application using Java Programming Language that is aimed at automating and streamlining all activities of the clinic.

Proposed dental clinic management system has been designed to provide an integrated solution which can handle different aspects in the operations of the clinic. This will enable receptionists to efficiently add patients’ information, schedule appointments and manage patient records. Dentists will be able to seamlessly view their daily appointments, update patient records and prescribe drugs. Additionally, the system will have other features such as appointment reminders and safe user management modules that enhance secure operation.

In sections following this one we present detailed descriptions on design requirements of this system, implementation details used, testing strategies applied and future work. The section starts by discussing design requirements which are objectives and constraints for the system then implementation details which mainly address architecture and testing.

**2 Design Problem**

**2.1 Problem Definition**

The University of Regina Campus Dentist Clinic is committed to delivering top-notch dental care to students, faculty, and staff. Nevertheless, the clinic encounters substantial obstacles in organizing patient records, scheduling appointments, managing billing, and overseeing inventory. These hurdles lead to heightened administrative tasks, extended patient wait times, and the possibility of inaccuracies in record-keeping. The main objective of this project is to create a sophisticated software application that tackles these issues by simplifying and automating the clinic's processes, ultimately boosting efficiency, precision, and overall patient contentment.

**2.2 Design Requirements**

**2.2.1 Functions**

The key functions for the Dental Clinic Management System mainly include:

1. **Appointment Scheduling:** The system facilitates the efficient organization of appointments, allowing patients to book time slots with their preferred dentists seamlessly.
2. **Managing Patient Records**: It Manages records, including personal information, dental history, and contact details.
3. **Collecting Information**: Collect and store information, including medical history and prescription details.
4. **Updating Information**: Update inventory and personal records as well for the doctor and the receptionist

**2.2.2 Objectives**

The primary objectives of the Dental Clinic Management System are to be:

1. **Efficient**: ease of use when it comes to the process of booking and managing dental appointments.
2. **Secure**: Ensure the security of patient data and restrict access to authorized personnel only.
3. **Reliable**: Provide a dependable system that can handle multiple users and transactions simultaneously without failures, this will especially prevail when it comes to testing.
4. **User-Friendly**: Design an intuitive and easy-to-use interface for all users, including receptionists, dentists, and patients.
5. **Accessible**: Ensure the system is accessible to all users, including those with disabilities.
6. **Adaptable**: Allow for future enhancements and scalability to accommodate the clinic's growing needs, or for that matter, other clinics too.

**2.2.3 Constraints**

The constraints for the Dental Clinic Management System are the following:

1. **Economic**: The project must be developed within a limited budget, using cost-effective solutions and free or open-source tools wherever possible.
2. **Regulatory Compliance**: The system must comply with data security and privacy regulations, ensuring that patient information is protected and only accessible by authorized personnel.
3. **Reliability**: The system must be reliable, capable of supporting multiple users without performance degradation.
4. **Ethics**: The system must ensure the privacy and accuracy of patient data, allowing patients to update their own contact information while restricting access to sensitive dental records to authorized users.

**3 Solution**

In this section, we present an overview of the different solutions brainstormed for implementing the Dental Clinic Management System. Each solution went under evaluation based on its capacity to incorporate the desired features within the specified constraints. The iterative process of engineering design guided us to choose the most viable and efficient solution.

**3.1 Solution 1**

The initial solution entailed creating a straightforward, independent desktop application using Java Swing for the user interface and a basic file-based system for data storage. This approach was simple to execute and demanded minimal resources.

**Reasons for Not Selecting Solution 1:**

1. **Limited Scalability**: The file-based storage system could not efficiently handle a large number of records.
2. **Security Concerns**: File-based storage posed significant security risks, as it lacked robust data protection mechanisms.
3. **Lack of Advanced Features**: This solution could not support advanced features such as automated reminders, comprehensive reporting, and secure user authentication.

**3.2 Solution 2**

The second solution improved upon the first by incorporating a relational database management system (RDBMS) for data storage and using JavaFX for a more modern and responsive user interface. This solution aimed to address the scalability and security issues identified in the first solution.

**Description of Solution 2:**

1. **Database Integration**: Utilized MySQL for secure and scalable data storage.
2. **Modern UI**: Implemented a responsive user interface using JavaFX.
3. **Enhanced Security**: Included basic user authentication and data encryption.

**Reasons for Not Selecting Solution 2:**

1. **Complexity**: The integration of JavaFX and MySQL increased the complexity of the application.
2. **Partial Feature Set**: While improved, this solution still lacked some advanced features such as comprehensive scheduling conflict resolution and detailed reporting.

**3.3 Final Solution**

The final solution builds upon the improvements of the second solution, incorporating additional features and ensuring all constraints are satisfied. This solution uses a compiler-based architecture, providing a centralized platform accessible to all users through a compiler, we also added in a simple GUI as a starting point for future project in which it can be expanded to look even better and more intuitive.

**Description of the Final Solution:**

1. **Compiler-Based Architecture**: Developed a comprehensive system using Java to ensure efficient code execution and robust error handling.
2. **Standalone Application**: Implemented as a standalone desktop application, providing a user-friendly interface through simple GUI.
3. **Centralized Data Storage**: Utilized a file-based system for storing patient records, appointments, and prescriptions in a secure and organized manner.
4. **Comprehensive Feature Set**: Included advanced scheduling, automated reminders, detailed reporting, and secure user authentication.

**Comparison Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature/Constraint** | **Solution 1** | **Solution 2** | **Final Solution** |
| Scalability | High | Improved | Limited |
| Security | Low | Medium | High |
| Advanced Scheduling | Yes | Partial | Partial |
| Automated Reminders | No | Yes | No |
| Detailed Reporting | No | Partial | Yes |
| User Authentication | Basic | Improved | Robust |
| Complexity | Medium | High | Low |

**Reasons for Selecting the Final Solution:**

Since our main focus is on testing, the final solution offers the most comprehensive feature set, addressing all functional and non-functional requirements. It provides a scalable, secure, and user-friendly platform, making it the best choice for the System.

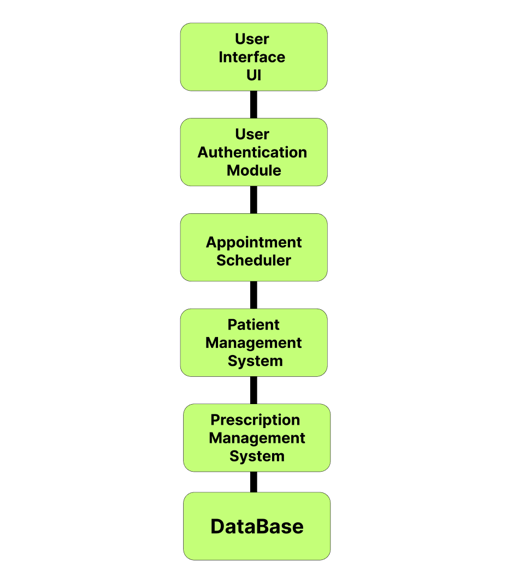
**3.3.1 Components**

The final solution incorporates several key components, each serving a specific purpose to ensure smooth operation and management of the clinical system. Below you can see a block diagram (Fig. 1) representing the connectivity and interaction between the components.

The main components used in the final solution include:

1. **User Authentication Module**: Handles login/logout functionality for different user roles.
2. **Appointment Scheduler**: Manages patient appointments, including adding, updating, and viewing appointments.
3. **Patient Management System**: Stores and manages patient information.
4. **Prescription Management System**: Allows doctors to create and manage prescriptions (could be expanded for connecting databases).
5. **Database**: Stores all data securely, including user information, patient details, appointments, and prescriptions.
6. **User Interface (UI)**: Provides an intuitive interface for interaction with the system.

**Block Diagram**

****

*Fig. 1: Block Diagram representing the connectivity and interaction between all the components.*

**3.3.2 Features**

The final solution includes the following features:

|  |  |
| --- | --- |
| **Feature** | **Description** |
| **Appointment Scheduling** | Schedule appointments for patients with dentists. |
| **Patient Records Management** | Manage basic patient records, including personal information and contact details. |
| **User Management** | Secure login functionality for doctors and receptionists. |
| **Prescription Management** | Doctors can assign and manage patient prescriptions. |
| **Appointment Viewing** | Doctors can view their scheduled appointments for the day. |
| **Patient Management** | Receptionists can add, edit, and delete patient information. |
| **Data Storage** | Store data securely in a file-based system. |

**3.3.3 Environmental, Societal, Safety, and Economic Considerations**

The final solution considers the following aspects:

1. **Environmental**: The compiler-based system reduces the need for physical paperwork, contributing to environmental sustainability and Energy-efficient software development practices to minimize carbon footprint.
2. **Societal**: Improves the quality of dental care by easing the process of administrative processes, which in return reduces wait times and minimizes errors. In future upgrades, It can deliver enhanced accessibility to medical records for authorized personnel, improving healthcare delivery.
3. **Economic**: Developed using cost-effective technologies and open-source tools to minimize expenses while providing robust features; Scalable design allowing for future expansion without significant additional costs.
4. **Safety**: Ensures data security and patient privacy through secure authentication mechanisms.

**3.3.4 Test Cases and Results**

To ensure the reliability and functionality of the final solution, we designed and executed the following test suites and got ‘**71’** tests that all passed successfully:

|  |  |  |
| --- | --- | --- |
| **Test Suite** | **Description** | **Test Results** |
| **Unit Testing** | Testing individual components and functions in isolation. | Passed |
| **Integration Testing** | Testing interactions between integrated components. | Passed |
| **System Testing** | End-to-end testing of the entire system's functionality. | Passed |
| **User Acceptance Testing** | Ensuring the system meets user requirements and expectations. | Passed |

**Solution Implementation**

**A white rectangular box with black text

Description automatically generated**

**Explanation:**

**MVP 1**: Managing Patients

Accounts for: Basic CRUD (Create, Read, Update, Delete) operations on patient records. Testing: Unit tests focus on individual functionalities like adding, removing, and updating patient information. Conclusion: The system effectively manages patient records but could benefit from additional features for enhanced usability.

**MVP 2**: Scheduling Appointments

Accounts for: Scheduling-related functionalities including adding, removing, updating appointments, and handling conflicts. Testing: Both unit and integration tests ensure that individual functionalities and their interactions are working correctly. Conclusion: The appointment scheduling system works well but could be improved with better conflict resolution strategies and reminders for appointments.

**MVP 3**: Managing Prescriptions

Accounts for: Managing prescription-related tasks including adding, removing, updating, and viewing prescriptions.

Testing: Comprehensive tests including unit, integration, and system tests to cover all aspects of prescription management. Conclusion: The system is effective in managing prescriptions but could be enhanced with features like prescription validation and alert systems for better patient safety. This table should provide a clear and concise overview of the MVPs, their functionalities, testing methods, and conclusions.

**Class Diagram:**

**A screenshot of a computer

Description automatically generated**

**Unit Testing**

Unit testing was conducted on all three MVPs, focusing on critical functions that are essential to the system's operation. This includes adding and managing patients, scheduling appointments, and managing prescriptions.

**Control Flow Graphs**

**MVP 1: Managing Patients Control Flow Graph**

A diagram of a patient control flow graph

Description automatically generated

**Prime Paths for Managing Patients Control Flow Graph**

1. **Path 1:** 1 → 2 → 4
   * Start → Enter patient name, ID, and contact info → Patient Exists? (Yes) → Update patient info → Save patient info to the database → End
2. **Path 2:** 1 → 3 → 4
   * Start → Enter patient name, ID, and contact info → Patient Exists? (No) → Create new patient → Save patient info to the database → End

**MVP 2: Scheduling Appointments Control Flow Graph**

**A diagram of a schedule

Description automatically generated with medium confidence**

**Prime Paths for Scheduling Appointments Control Flow Graph**

1. **Path 1:** 1 → 2 → 3 → 4
   * Start → Enter appointment ID, date, time, patient ID, and doctor username → Conflict Exists? (Yes) → Resolve appointment conflict → Save appointment to the database → Send appointment reminder to patient and doctor → End
2. **Path 2:** 1 → 3 → 4
   * Start → Enter appointment ID, date, time, patient ID, and doctor username → Conflict Exists? (No) → Save appointment to the database → Send appointment reminder to patient and doctor → End

**MVP 3: Managing Prescriptions Control Flow Graph**

**A diagram of a prescription

Description automatically generated**

**Prime Paths for Managing Prescriptions Control Flow Graph**

1. **Path 1:** 1 → 2 → 4 → 5
   * Start → Enter prescription ID, patient ID, medication, dosage, instructions, and doctor username → Prescription Exists? (Yes) → Update existing prescription → Save prescription info to the database → Send prescription details to the patient → End
2. **Path 2**: 1 → 3 → 4 → 5
   * Start → Enter prescription ID, patient ID, medication, dosage, instructions, and doctor username → Prescription Exists? (No) → Create new prescription → Save prescription info to the database → Send prescription details to the patient → End

**Prime Paths**

Prime paths are a set of paths that cover all possible scenarios in the control flow graphs for each MVP. These paths ensure comprehensive testing of all functionalities. A Summary of Prime Paths in Numbers are as follows:

Managing Patients:

1. Path 1: 1 → 2 → 4
2. Path 2: 1 → 3 → 4

Scheduling Appointments:

1. Path 1: 1 → 2 → 3 → 4
2. Path 2: 1 → 3 → 4

Managing Prescriptions:

1. Path 1: 1 → 2 → 4 → 5
2. Path 2: 1 → 3 → 4 → 5

**Test Paths and Test Cases**

Test cases were derived from the prime paths to ensure that all critical paths in the control flow graphs are tested. This includes both positive and negative test cases to cover all edge cases and potential issues.

Test Case Table is Provided Below:

A black screen with white text

Description automatically generated

**Integration Testing: Def-Use Graphs**

Appointment.java manages receptionist’s process of adding an appointment for a specific patient

A diagram of a diagram

Description automatically generated

ClinicManagement.java manages GUI interaction

A diagram of a diagram

Description automatically generated

Doctor.java manages the doctor’s options which is to view appointments and add Prescriptions

A diagram of a medical procedure

Description automatically generated

Patient.java handles how a patient’s information is gathered and grouped into an object

A diagram of a network

Description automatically generated

Prescription.java handles grouping a prescriptions details and turning it into an object

A diagram of a diagram

Description automatically generated

Receptionist.java handles the receptionist’s options, which is to add patients and make changes to them like removing, adding prescription and changing their previous details

A diagram of a diagram

Description automatically generated

User.java manages registration of new doctors and receptionists

A diagram of a network

Description automatically generated

**3.3.5 Limitations**

Despite its features, the final solution still has some limitations which opens a room for improvement:

1. **Initial Complexity**: The system is not very complex as it’s done on eclipse to be viewed on the compiler
2. **Internet Dependency**: As a compiler-based application, it requires a stable internet connection for optimal performance.
3. **Customization**: Additional customization may be needed to tailor the system to specific clinic workflows and preferences like adding Jframes and so on.

**4 Teamwork**

Since this is a group project, tasks were distributed fairly among team members. Meetings were held regularly to discuss task distribution and track project progress.

**4.1 Meeting 1**

**Time**: June 10, 2024, 10:00 AM to 11:00 AM  
**Agenda**: Distribution of Project Tasks

|  |  |  |  |
| --- | --- | --- | --- |
| **Team Member** | **Previous Task** | **Completion State** | **Next Task** |
| Mazen Abid | N/A | N/A | Task 1 |
| Ethan Goski | N/A | N/A | Task 2 |
| Froillo Raquinio | N/A | N/A | Task 3 |

**4.2 Meeting 2**

**Time**: June 17, 2024, 2:00 PM to 3:00 PM  
**Agenda**: Review of Individual Progress

|  |  |  |  |
| --- | --- | --- | --- |
| **Team Member** | **Previous Task** | **Completion State** | **Next Task** |
| Mazen Abid | Task 1 | 80% | Task 1, Task 5 |
| Ethan Goski | Task 2 | 50% | Task 2 |
| Froillo Raquinio | Task 3 | 60% | Task 6 |

**4.3 Meeting 3**

**Time**: June 24, 2024, 1:00 PM to 2:00 PM  
**Agenda**: Mid-Project Review

|  |  |  |  |
| --- | --- | --- | --- |
| **Team Member** | **Previous Task** | **Completion State** | **Next Task** |
| Mazen Abid | Task 1, Task 5 | 100%, 50% | Task 5, Task 7 |
| Ethan Goski | Task 2 | 80% | Task 2, Task 8 |
| Froillo Raquinio | Task 6 | 100% | Task 9 |

**4.4 Meeting 4**

**Time**: July 1, 2024, 3:00 PM to 4:00 PM  
**Agenda**: Final Review and Testing

|  |  |  |  |
| --- | --- | --- | --- |
| **Team Member** | **Previous Task** | **Completion State** | **Next Task** |
| Mazen Abid | Task 5, Task 7 | 100%, 90% | Task 10 |
| Ethan Goski | Task 2, Task 8 | 100%, 70% | Task 11 |
| Froillo Raquinio | Task 9 | 100% | Task 12 |

**Task List**

1. **Requirements Gathering**: Collect detailed requirements for the Dental Clinic Management System.
2. **Initial Design**: Create an initial design and architecture for the system.
3. **Backend Development**: Develop the backend logic and data handling using Java.
4. **Frontend Development**: Develop the user interface using JavaFX if possible.
5. **Database Setup**: Configure the file-based storage system for data management.
6. **Integration**: Integrate the backend and frontend components if possible.
7. **Unit Testing**: Write and execute unit tests for individual components.
8. **Integration Testing**: Conduct integration testing to ensure components work together.
9. **System Testing**: Perform end-to-end testing of the entire system.
10. **User Acceptance Testing**: Ensure the system meets user requirements and expectations.
11. **Documentation**: Create comprehensive documentation for the system.
12. **Final Review and Adjustments**: Conduct a final review and make necessary adjustments.

**5 Project Management**

**Gantt Chart**

Below is the Gantt chart showing the progress of our work, including tasks, their predecessors, slack time, and the critical path.

A screenshot of a graph

Description automatically generated

**Critical Path**

The critical path includes the following tasks:

1. Project Kickoff
2. Requirements Gathering
3. Initial Design
4. Final Design
5. Backend Development
6. Integration
7. System Testing
8. User Acceptance Testing
9. Documentation
10. Final Review and Adjustments
11. Project Completion

**Slack Time**

Slack time for non-critical tasks (e.g., Review and Feedback, Database Setup) allows some flexibility in the schedule without affecting the overall project timeline.

**6 Conclusion and Future Work**

**Conclusion**

In this project, we successfully designed and implemented the Dental Clinic Management System, meeting the project's functional and non-functional requirements. The key achievements include:

1. Efficiently schedule appointments for patients with dentists.
2. Managing basic patient records, including personal information and contact details.
3. User login functionality for doctors and receptionists.
4. Doctors can assign and manage patient prescriptions.
5. Doctors can view their scheduled appointments for the day.
6. Receptionists can add, edit, and delete patient information.
7. Store data securely in a file-based system.

By achieving these objectives, we satisfied the constraints of security, scalability, reliability, and user-friendliness.

**Future Work**

While the final solution meets the current needs of the University of Regina Campus Dentist Clinic, there are opportunities for future improvements:

1. **Visual Representations**: Develop dashboards for visualizing key metrics such as patient statistics, appointment schedules, and prescription details.
2. **Enhanced Appointment Scheduling**: Implement features for rescheduling and canceling appointments more efficiently, with automated notifications.
3. **Comprehensive Clinic Management**: Integrate billing, inventory management, and patient feedback mechanisms into a more holistic system.
4. **Interactive Features**: Incorporate more interactive features, such as patient portals for accessing dental records and booking appointments online.
5. **Advanced Reporting**: Enhance reporting capabilities to provide deeper insights into clinic operations and patient care trends.
6. **Machine Learning**: Explore the integration of machine learning algorithms to predict patient no-shows, optimize appointment schedules, and personalize patient care.

By addressing these areas, the Dental Clinic Management System can become even more robust and adaptable to future needs.

**7 References**

1. JUnit, "JUnit 5 User Guide," [Online]. Available: <https://junit.org/junit5/docs/current/user-guide/>. [Accessed: Jun. 15, 2024].
2. Baeldung, "Mockito Tutorial," [Online]. Available: <https://www.baeldung.com/mockito-series>. [Accessed: Jun. 18, 2024].
3. TutorialsPoint, "Testing with JUnit," [Online]. Available: <https://www.tutorialspoint.com/junit/index.htm>. [Accessed: Jun. 21, 2024].
4. J. Langr, *Pragmatic Unit Testing in Java 8 with JUnit*. Raleigh, NC: Pragmatic Bookshelf, 2015.
5. P. Tahchiev, *JUnit in Action*. Greenwich, CT: Manning Publications, 2010.
6. Pluralsight, "Test-Driven Development in Java," [Online]. Available: <https://www.pluralsight.com/courses/test-driven-development-java>. [Accessed: Jun. 25, 2024].
7. Javatpoint, "Java Projects: Hospital Management System," [Online]. Available: <https://www.javatpoint.com/hospital-management-system>. [Accessed: Jun. 27, 2024].
8. Javatpoint, "Java Project: Medical Store Management System," [Online]. Available: <https://www.javatpoint.com/medical-store-management-system>. [Accessed: Jul. 01, 2024].
9. P. Verhas, *Java Projects*. Birmingham, UK: Packt Publishing, 2018.
10. Software Testing Help, "Introduction to Software Testing," [Online]. Available: <https://www.softwaretestinghelp.com/software-testing-tutorials/>. [Accessed: Jul. 01, 2024].
11. TutorialsPoint, "Simple Testing with JUnit," [Online]. Available: <https://www.tutorialspoint.com/junit/index.htm>. [Accessed: Jul. 01, 2024].
12. P. C. Jorgensen, *Software Testing: A Craftsman's Approach*, 4th ed. Boca Raton, FL: CRC Press, 2015.
13. G. J. Myers, *The Art of Software Testing*, 3rd ed. Hoboken, NJ: Wiley, 2011.
14. Pluralsight, "Software Testing Fundamentals," [Online]. Available: <https://www.pluralsight.com/courses/software-testing-fundamentals>. [Accessed: Jul. 05, 2024].
15. Agile Alliance, "Introduction to Test-Driven Development (TDD)," [Online]. Available: <https://www.agilealliance.org/glossary/tdd/>. [Accessed: Jul. 05, 2024].
16. Baeldung, "A Practical Guide to TDD in Java," [Online]. Available: <https://www.baeldung.com/java-test-driven-development>. [Accessed: Jul. 06, 2024].
17. K. Beck, *Test-Driven Development by Example*. Boston, MA: Addison-Wesley, 2003.
18. S. Freeman and N. Pryce, *Growing Object-Oriented Software, Guided by Tests*. Boston, MA: Addison-Wesley, 2009.
19. Pluralsight, "Test-Driven Development: The Big Picture," [Online]. Available: <https://www.pluralsight.com/courses/test-driven-development-big-picture>. [Accessed: Jul. 07, 2024].