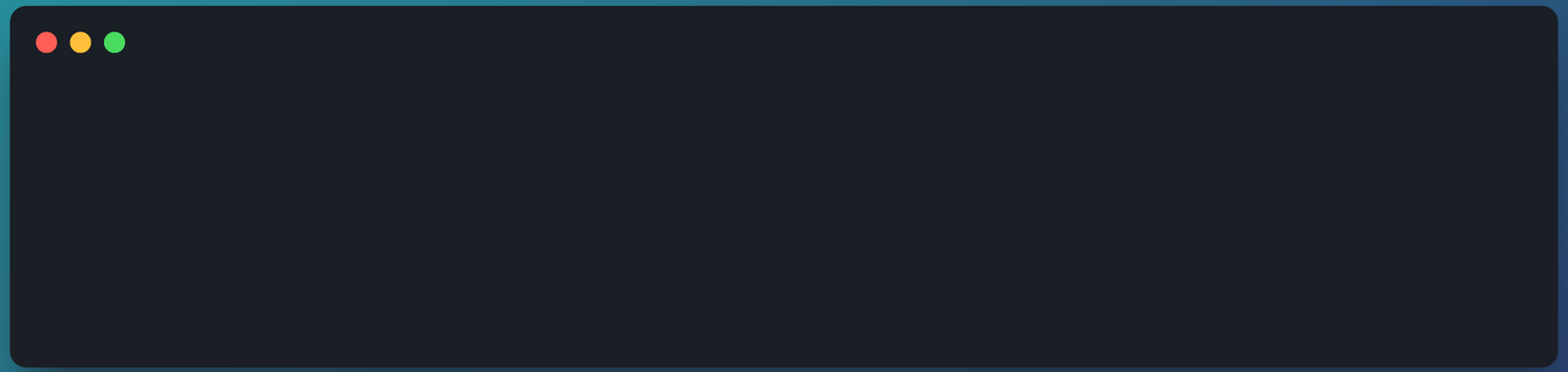
Exceptions: Miscellaneous topics.

```
/*
    . Standard expceptions:
        .#1: Catching standard exceptions

        .#2: Throwing standard exceptions

        .#3: Deriving from standard exceptions

        .#4: File I/O with exceptions
            . Comprison with return value based error handling

*/
```

Exceptions: Catching standard exceptions

```
Animal animal;    // Animal can't bee cast to Feline.
try {
    Feline &feline_ref = dynamic_cast<Feline &>(animal); // will throw std::bad_cast
} catch (std::exception &ex) {
    fmt::println("Something is wrong : {}", ex.what());
```

Exceptions: Catching standard exceptions - Demo time!



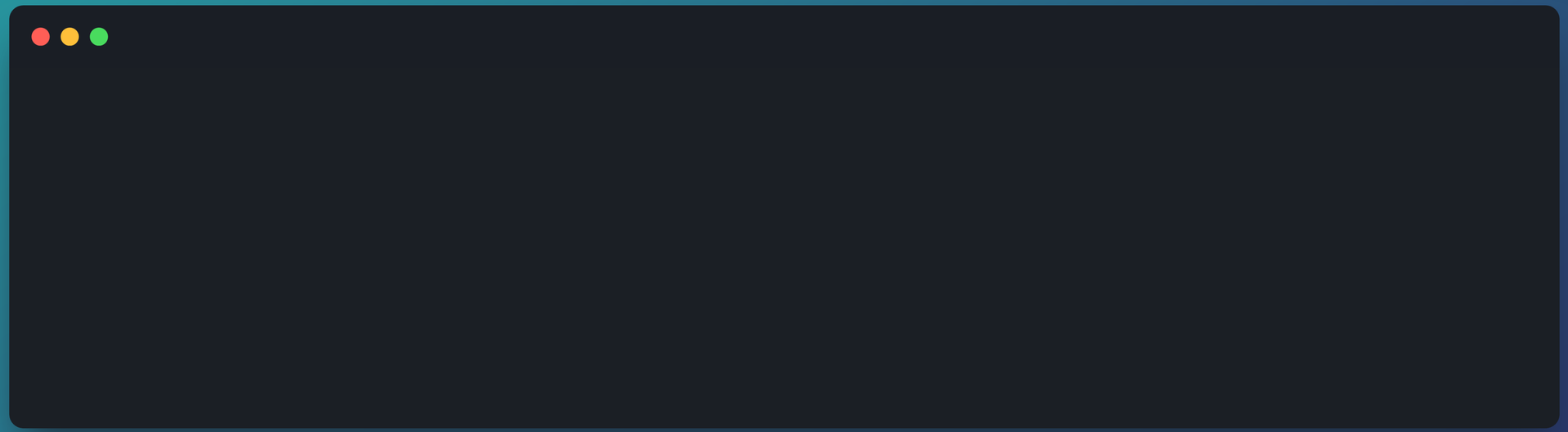
Exceptions: Throwing standard exceptions.

```cpp
export class Students
{
public:
    Students() = delete;
    Students(std::string_view s1, std::string_view s2
                , std::string_view s3, std::string_view s4, std::string_view s5)
    {
        m_students[0] = s1;
        m_students[1] = s2;
        m_students[2] = s3;
        m_students[3] = s4;
        m_students[4] = s5;
    }
    ~Students() = default;

    std::string_view get_student(size_t index)
    {
        const std::string message = "Index out of range, valid range["
                                        + std::to_string(0) + "," + std::to_string(4) + "]";

        if ((index < 0) || (index >= 5)) throw std::out_of_range(message);
        return m_students[index];
    }

private:
    std::string m_students[5];
};
```

Exceptions: Throwing standard exceptions.

```cpp
exceptions_2::Students students("John Snow", "Samuel Njuguna", "Nicholai Itchenko", "Bilom Atunde", "Lily Park");

try {
    // fmt::println( students.get_student(2) );
    fmt::println("{}", students.get_student(-2));
} catch (std::exception &ex) {
    fmt::println("Exception cought : {}", ex.what());
}
```

Exceptions: Throwing standard exceptions - Demo time!



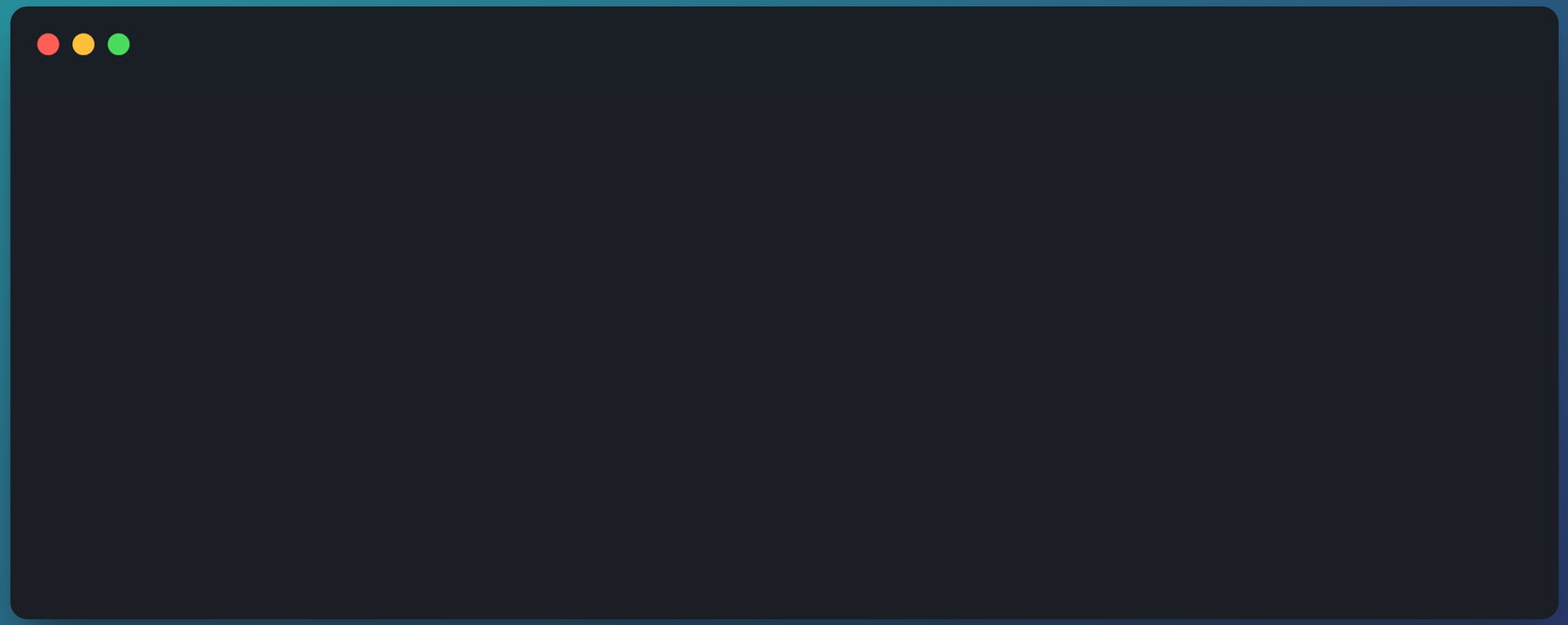
Exceptions: Deriving from standard exceptions

```
export class ArrayIndexOutOfRangeException : public std::exception {
public:
    ArrayIndexOutOfRangeException(int index, int max_size) noexcept
        : m_index(index), m_max_size(max_size),
        m_message("Error: Index " + std::to_string(m_index) +
                  " out of range (valid range: 0 to " + std::to_string(m_max_size - 1) + ")") {}

    const char* what() const noexcept override {
        return m_message.c_str();
    }

    int get_index() const { return m_index; }
    int get_max_size() const { return m_max_size; }

private:
    int m_index;
    int m_max_size;
    std::string m_message;
};


// Template function to access elements safely from a std::array
export template <typename T, std::size_t N>
T access_element(const std::array<T, N>& arr, int index) {
    if (index < 0 || index ⩾ static_cast<int>(N)) {
        throw ArrayIndexOutOfRangeException(index, N);
    }
    return arr[index];
}
```

Exceptions: Deriving from standard exceptions

```
std::array<int, 5> arr = {10, 20, 30, 40, 50};

try {
    // This will throw an exception as we attempt to access an invalid index (e.g., 10)
    int value = access_element(arr, 10);
    fmt::println("Element at index 10: {}", value);
} catch (const ArrayIndexOutOfRangeException& e) {
    fmt::println("Caught exception: {}", e.what());
}
```

Exceptions: Deriving from standard exceptions - Demo time!



Exceptions: File I/O - An example.

```
export void write_csv(const std::filesystem::path& file_path) {
    // Create a CSV file with some data
    std::ofstream csv_file(file_path, std::ios::out);
    if (!csv_file) {
        fmt::println("Failed to create file: {}", file_path.string());
        return;
    }
    //Write operations

    csv_file.close();
    fmt::println("CSV file written successfully to: {}", file_path.string());
}

export void read_csv(const std::filesystem::path& file_path) {
    // Check if the file exists using std::filesystem
    if (!std::filesystem::exists(file_path)) {
        fmt::println("File does not exist: {}", file_path.string());
        return;
    }

    std::ifstream csv_file(file_path);
    if (!csv_file) {
        fmt::println("Failed to open file: {}", file_path.string());
        return;
    }
    //Read operations

    csv_file.close();
}
```

Exceptions: File I/O - An example.

```
    // Create a CSV file with some data
    std::ofstream csv_file(file_path, std::ios::out);
    if (!csv_file) {
        throw std::runtime_error(fmt::format("Failed to create file: {}", file_path.string()));
    }

    csv_file << "Name, Age, Occupation\n";
    csv_file << "Alice, 30, Engineer\n";
    csv_file << "Bob, 25, Designer\n";
    csv_file << "Charlie, 35, Teacher\n";

    csv_file.close();
    fmt::println("CSV file written successfully to: {}", file_path.string());
}
```

Exceptions: File I/O - An example.

```cpp
export void read_csv(const std::filesystem::path& file_path) {
    // Check if the file exists using std::filesystem
    if (!std::filesystem::exists(file_path)) {
        throw std::runtime_error(fmt::format("File does not exist: {}", file_path.string()));
    }

    std::ifstream csv_file(file_path);
    if (!csv_file) {
        throw std::runtime_error(fmt::format("Failed to open file: {}", file_path.string()));
    }

    std::string line;
    while (std::getline(csv_file, line)) {
        fmt::println("{}", line);
    }

    csv_file.close();
}
```

Exceptions: File I/O - An example.

```
{
    //Without exceptions
    std::filesystem::path file_path = R"(D:\sample_file.csv)"; // Windows
    //std::filesystem::path file_path = R"(/path/to/your/sample_file.csv)"; // Linux

    using exceptions_4::csv_no_except::write_csv;
    using exceptions_4::csv_no_except::read_csv;

    //Write and read the file
    write_csv(file_path);
    read_csv(file_path);
}
```

Exceptions: File I/O - An example.

```
{
  //With exceptions
  std::filesystem::path file_path = R"(D:\sample_file.csv)"; // Windows
  //std::filesystem::path file_path = R"(/path/to/your/sample_file.csv)"; // Linux

  using exceptions_4::csv_with_except::write_csv;
  using exceptions_4::csv_with_except::read_csv;

  //Write and read the file
  try{
    write_csv(file_path);
  } catch (std::exception &ex) {
    fmt::println("Exception cought : {}", ex.what());
  }

  try{
    read_csv(file_path);
  } catch (std::exception &ex) {
    fmt::println("Exception cought : {}", ex.what());
  }
}
```

Exceptions: File I/O - An example - Demo time!