Cambridge Academy for Science and Technology

AQA Computer Science

Practical Computing Project

CRYPTOGRAPHY ONLINE

Author
J.P. Jacob Powell

Supervisor
B.C. Barry Cooper

February 1, 2018

Contents

Chapter 1

Analysis

1.1 What is Cryptology

Chapter 2

Documented Design

2.1 High-Level Overview

2.2 Description of Algorithms

2.3 Description of Data Structures

2.4 Design of Database

2.5 Design of user interface

2.6 Hardware/Software Selection

2.7 System Security and Integrity of Data

Chapter 3

Technical Solution

3.1 Testing Code Input

Here is some code.

```cpp
/*
 * File: crypto_online_home.cc
 * Created: 19/12/2017 13:32
 * Finished:
 *
 * Description:
 *
 * Author: Jacob Powell
 */

#include "crypto_online_navigation_grid.h"

#include <Wt/WPushButton.h>

crypto_online_navigation_grid::crypto_online_navigation_grid() {
    populate_navigation_grid();
}


void crypto_online_navigation_grid::populate_navigation_grid() {
    this->setup_basic_contents();
    this->setup_symmetric_contents();
    this->setup_asymmetric_contents();
    this->setup_applications_contents();
    this->clear_grid();
}

void crypto_online_navigation_grid::setup_basic_contents() {
    this->title_cryptography_basics = Wt::cpp14::make_unique<Wt::WText>("Cryptography Basics");
    this->title_cryptography_basics->setStyleClass("navigation_grid_title");
    this->addWidget(std::move(this->title_cryptography_basics),0,1);

    this->cryptography_basics_concepts = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/basics/concepts"));
    auto button_cryptography_basics_concepts = Wt::cpp14::make_unique<Wt::WPushButton>("Basic Concepts");
    button_cryptography_basics_concepts->setStyleClass("navigation_grid_item");
    this->cryptography_basics_concepts->addWidget(std::move(button_cryptography_basics_concepts));
    this->addWidget(std::move(cryptography_basics_concepts),1,0, Wt::AlignmentFlag::Center);

    this->cryptography_basics_real_world_applications = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/basics/real-world-applications"));
    auto button_cryptography_basics_real_world_applications =
            Wt::cpp14::make_unique<Wt::WPushButton>("Real World Applications");
    button_cryptography_basics_real_world_applications->setStyleClass("navigation_grid_item");
    this->cryptography_basics_real_world_applications->addWidget(
            std::move(button_cryptography_basics_real_world_applications));
    this->addWidget(std::move(this->cryptography_basics_real_world_applications),1,1, Wt::AlignmentFlag::Center);

    this->cryptography_basics_concepts_history = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/basics/history"));
    auto button_cryptography_basics_history = Wt::cpp14::make_unique<Wt::WPushButton>("History");
    button_cryptography_basics_history->setStyleClass("navigation_grid_item");
    this->cryptography_basics_concepts_history->addWidget(
            std::move(button_cryptography_basics_history));
    this->addWidget(std::move(this->cryptography_basics_concepts_history),1,2, Wt::AlignmentFlag::Center);
}

void crypto_online_navigation_grid::setup_symmetric_contents() {
    this->title_symmetric = Wt::cpp14::make_unique<Wt::WText>("Symmetric Cryptography");
    this->title_symmetric->setStyleClass("navigation_grid_title");
    this->addWidget(std::move(this->title_symmetric),2,1);

    this->subtitle_introduction_to_symmetric_cryptography = Wt::cpp14::make_unique<Wt::WText>("Introduction to Symmetric Cryptography");
    this->subtitle_introduction_to_symmetric_cryptography->setStyleClass("navigation_grid_subtitle");
    this->addWidget(std::move(this->subtitle_introduction_to_symmetric_cryptography),3,1);

    this->symmetric_modular_arithmetic = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/symmetric/modular-arithmetic"));
    auto button_cryptography_symmetric_modular_arithmetic = Wt::cpp14::make_unique<Wt::WPushButton>("Modular Arithmetic");
    button_cryptography_symmetric_modular_arithmetic->setStyleClass("
navigation_grid_item");
    this->symmetric_modular_arithmetic->addWidget(
            std::move(button_cryptography_symmetric_modular_arithmetic));
    this->addWidget(std::move(this->symmetric_modular_arithmetic),4,0, Wt::AlignmentFlag::Center);

    this->symmetric_cipher_types = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/symmetric/cipher-types"));
    auto button_cryptography_symmetric_cipher_types = Wt::cpp14::make_unique<Wt::WPushButton>("Cipher Types");
    button_cryptography_symmetric_cipher_types->setStyleClass("navigation_grid_item");
    this->symmetric_cipher_types->addWidget(std::move(button_cryptography_symmetric_cipher_types));
    this->addWidget(std::move(this->symmetric_cipher_types),4,1, Wt::AlignmentFlag::Center);

    this->symmetric_historical_ciphers = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/symmetric/historical-ciphers"));
    auto button_cryptography_symmetric_historical_ciphers = Wt::cpp14::make_unique<Wt::WPushButton>("Historical Ciphers");
    button_cryptography_symmetric_historical_ciphers->setStyleClass("navigation_grid_item");
    this->symmetric_historical_ciphers->addWidget(std::move(button_cryptography_symmetric_historical_ciphers));
    this->addWidget(std::move(this->symmetric_historical_ciphers),4,2, Wt::AlignmentFlag::Center);

    this->symmetric_randomness = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/symmetric/random-number-generators"));
    auto button_cryptography_symmetric_randomness = Wt::cpp14::make_unique<Wt::WPushButton>("Random Number Generators");
    button_cryptography_symmetric_randomness->setStyleClass("navigation_grid_item");
    this->symmetric_randomness->addWidget(std::move(button_cryptography_symmetric_randomness));
    this->addWidget(std::move(this->symmetric_randomness),5,0, Wt::AlignmentFlag::Center);

    this->symmetric_stream_ciphers = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/symmetric/stream-ciphers"));
    auto button_cryptography_symmetric_stream_ciphers = Wt::cpp14::make_unique<Wt::WPushButton>("Stream Ciphers");
    button_cryptography_symmetric_stream_ciphers->setStyleClass("navigation_grid_item");
    this->symmetric_stream_ciphers->addWidget(std::move(button_cryptography_symmetric_stream_ciphers));
    this->addWidget(std::move(this->symmetric_stream_ciphers),5,1, Wt::AlignmentFlag::Center);

    this->symmetric_linear_feedback_shift_registers = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/symmetric/stream-ciphers"));
    auto button_cryptography_symmetric_lfsr = Wt::cpp14::make_unique<Wt::WPushButton>("Linear Feedback Shift Registers");
    button_cryptography_symmetric_lfsr->setStyleClass("navigation_grid_item");
    this->symmetric_linear_feedback_shift_registers->addWidget(std::move(button_cryptography_symmetric_lfsr));
    this->addWidget(std::move(this->symmetric_linear_feedback_shift_registers),5,2, Wt::AlignmentFlag::Center);

    this->subtitle_des = Wt::cpp14::make_unique<Wt::WText>("DES: Data Encryption Standard");
    this->subtitle_des->setStyleClass("navigation_grid_subtitle");
    this->addWidget(std::move(subtitle_des),6,1);

    this->symmetric_des_overview = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/symmetric/des-overview"));
    auto button_cryptography_symmetric_des_overview = Wt::cpp14::make_unique<Wt::WPushButton>("DES Overview");
    button_cryptography_symmetric_des_overview->setStyleClass("navigation_grid_item");
    this->symmetric_des_overview->addWidget(std::move(button_cryptography_symmetric_des_overview));
    this->addWidget(std::move(this->symmetric_des_overview),7,0, Wt::AlignmentFlag::Center);

    this->symmetric_des_feistal_networks = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/symmetric/feistal-networks"));
    auto button_cryptography_symmetric_des_feistal_networks = Wt::cpp14::make_unique<Wt::WPushButton>("Feistal Networks");
    button_cryptography_symmetric_des_feistal_networks->setStyleClass("navigation_grid_item");
    this->symmetric_des_feistal_networks->addWidget(std::move(button_cryptography_symmetric_des_feistal_networks));
    this->addWidget(std::move(this->symmetric_des_feistal_networks),7,1, Wt::AlignmentFlag::Center);

    this->symmetric_des_internals = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/symmetric/des-internals"));
    auto button_cryptography_symmetric_des_internals = Wt::cpp14::make_unique<Wt::WPushButton>("DES Internals");
    button_cryptography_symmetric_des_internals->setStyleClass("navigation_grid_item");
    this->symmetric_des_internals->addWidget(std::move(button_cryptography_symmetric_des_internals));
    this->addWidget(std::move(this->symmetric_des_internals),7,2, Wt::AlignmentFlag::Center);

    this->symmetric_des_decryption = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/symmetric/des-decryption"));
    auto button_cryptography_symmetric_des_decryption = Wt::cpp14::make_unique<Wt::WPushButton>("DES Decryption");
    button_cryptography_symmetric_des_decryption->setStyleClass("
        navigation_grid_item");
    this->symmetric_des_decryption->addWidget(std::move(↩
        button_cryptography_symmetric_des_decryption));
    this->addWidget(std::move(this->symmetric_des_decryption),8,0, Wt::↩
        AlignmentFlag::Center);

    this->symmetric_des_security = Wt::cpp14::make_unique<Wt::WAnchor>(Wt↩
        ::WLink(
             Wt::LinkType::InternalPath, "/symmetric/des-security"));
    auto button_cryptography_symmetric_des_security = Wt::cpp14::↩
        make_unique<Wt::WPushButton>("DES Security");
    button_cryptography_symmetric_des_security->setStyleClass("↩
        navigation_grid_item");
    this->symmetric_des_security->addWidget(std::move(↩
        button_cryptography_symmetric_des_security));
    this->addWidget(std::move(this->symmetric_des_security),8,1, Wt::↩
        AlignmentFlag::Center);

    this->symmetric_des_alternatives = Wt::cpp14::make_unique<Wt::WAnchor↩
        >(Wt::WLink(
             Wt::LinkType::InternalPath, "/symmetric/des-alternatives"));
    auto button_cryptography_symmetric_des_alternatives = Wt::cpp14::↩
        make_unique<Wt::WPushButton>("DES Alternatives");
    button_cryptography_symmetric_des_alternatives->setStyleClass("↩
        navigation_grid_item");
    this->symmetric_des_alternatives->addWidget(std::move(↩
        button_cryptography_symmetric_des_alternatives));
    this->addWidget(std::move(this->symmetric_des_alternatives),8,2, Wt::↩
        AlignmentFlag::Center);

    this->subtitle_aes = Wt::cpp14::make_unique<Wt::WText>("AES: Advanced↩
         Encryption Standard");
    this->subtitle_aes->setStyleClass("navigation_grid_subtitle");
    this->addWidget(std::move(subtitle_aes),9,1);

    this->symmetric_aes_overview = Wt::cpp14::make_unique<Wt::WAnchor>(Wt↩
        ::WLink(
             Wt::LinkType::InternalPath, "/symmetric/aes-overview"));
    auto button_cryptography_symmetric_aes_overview = Wt::cpp14::↩
        make_unique<Wt::WPushButton>("AES Overview");
    button_cryptography_symmetric_aes_overview->setStyleClass("↩
        navigation_grid_item");
    this->symmetric_aes_overview->addWidget(std::move(↩
        button_cryptography_symmetric_aes_overview));
    this->addWidget(std::move(this->symmetric_aes_overview),10,0, Wt::↩
        AlignmentFlag::Center);

    this->symmetric_aes_galois_fields = Wt::cpp14::make_unique<Wt::↩
        WAnchor>(Wt::WLink(
             Wt::LinkType::InternalPath, "/symmetric/aes-galois-fields"));
    auto button_cryptography_symmetric_aes_galois_fields = Wt::cpp14::↩
        make_unique<Wt::WPushButton>("AES Galois Fields");
    button_cryptography_symmetric_aes_galois_fields->setStyleClass("↩
        navigation_grid_item");
    this->symmetric_aes_galois_fields->addWidget(std::move(↩
        button_cryptography_symmetric_aes_galois_fields));
    this->addWidget(std::move(this->symmetric_aes_galois_fields),10,1, Wt↩
        ::AlignmentFlag::Center);

    this->symmetric_aes_internals = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/symmetric/aes-internals"));
    auto button_cryptography_symmetric_aes_internals = Wt::cpp14::make_unique<Wt::WPushButton>("AES Internals");
    button_cryptography_symmetric_aes_internals->setStyleClass("navigation_grid_item");
    this->symmetric_aes_internals->addWidget(std::move(button_cryptography_symmetric_aes_internals));
    this->addWidget(std::move(this->symmetric_aes_internals),10,2, Wt::AlignmentFlag::Center);

    this->symmetric_aes_decryption = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/symmetric/aes-decryption"));
    auto button_cryptography_symmetric_aes_decryption = Wt::cpp14::make_unique<Wt::WPushButton>("AES Decryption");
    button_cryptography_symmetric_aes_decryption->setStyleClass("navigation_grid_item");
    this->symmetric_aes_decryption->addWidget(std::move(button_cryptography_symmetric_aes_decryption));
    this->addWidget(std::move(this->symmetric_aes_decryption),11,0, Wt::AlignmentFlag::Center);

    this->symmetric_aes_implementations_hardware_software = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/symmetric/aes-implementations-hardware-software"));
    auto button_cryptography_symmetric_aes_implementations =
            Wt::cpp14::make_unique<Wt::WPushButton>("AES Implementations in Hardware and Software");
    button_cryptography_symmetric_aes_implementations->setStyleClass("navigation_grid_item");
    this->symmetric_aes_implementations_hardware_software->addWidget(std::move(button_cryptography_symmetric_aes_implementations));
    this->addWidget(std::move(this->symmetric_aes_implementations_hardware_software),11,1, Wt::AlignmentFlag::Center);

    this->symmetric_aes_example = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/symmetric/aes-example"));
    auto button_cryptography_symmetric_aes_example = Wt::cpp14::make_unique<Wt::WPushButton>("AES Example");
    button_cryptography_symmetric_aes_example->setStyleClass("navigation_grid_item");
    this->symmetric_aes_example->addWidget(std::move(button_cryptography_symmetric_aes_example));
    this->addWidget(std::move(this->symmetric_aes_example),11,2, Wt::AlignmentFlag::Center);

    this->subtitle_more_about_block_ciphers = Wt::cpp14::make_unique<Wt::WText>("More about Block Ciphers");
    this->subtitle_more_about_block_ciphers->setStyleClass("navigation_grid_subtitle");
    this->addWidget(std::move(this->subtitle_more_about_block_ciphers),12,1);

    this->symmetric_more_about_block_ciphers_modes_of_operation = Wt::↩
        cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/symmetric/modes-of-operation"))↩
                ;
    auto ↩
        button_cryptography_symmetric_more_about_block_ciphers_modes_of_operation↩
         =
            Wt::cpp14::make_unique<Wt::WPushButton>("Modes of Operation")↩
                ;
    button_cryptography_symmetric_more_about_block_ciphers_modes_of_operation↩
        ->setStyleClass("navigation_grid_item");
    this->symmetric_more_about_block_ciphers_modes_of_operation->↩
        addWidget(
            std::move(↩
                button_cryptography_symmetric_more_about_block_ciphers_modes_of_operatio
                ));
    this->addWidget(std::move(this->↩
        symmetric_more_about_block_ciphers_modes_of_operation),13,0, Wt::↩
        AlignmentFlag::Center);

    this->symmetric_more_about_block_ciphers_increasing_security = Wt::↩
        cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/symmetric/increasing-block-↩
                cipher-security"));
    auto ↩
        button_cryptography_symmetric_more_about_block_ciphers_increasing_security↩
         =
            Wt::cpp14::make_unique<Wt::WPushButton>("Increasing security ↩
                of Block Ciphers");
    button_cryptography_symmetric_more_about_block_ciphers_increasing_security↩
         ->setStyleClass("navigation_grid_item");
    this->symmetric_more_about_block_ciphers_increasing_security->↩
        addWidget(
            std::move(↩
                button_cryptography_symmetric_more_about_block_ciphers_increasing_securi
                ));
    this->addWidget(std::move(this->↩
        symmetric_more_about_block_ciphers_increasing_security),13,1, Wt::↩
        AlignmentFlag::Center);

    this->↩
        symmetric_more_about_block_ciphers_revisit_of_exhaustive_key_search↩
         = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/symmetric/revisit-of-exhaustive↩
                -key-search"));
    auto ↩
        button_cryptography_symmetric_more_about_block_ciphers_revisit_of_exhaustive_key
         =
            Wt::cpp14::make_unique<Wt::WPushButton>("Revisit of ↩
                Exhaustive Key Search");
    button_cryptography_symmetric_more_about_block_ciphers_revisit_of_exhaustive_key_sea
        ->setStyleClass("navigation_grid_item");
    this->↩
        symmetric_more_about_block_ciphers_revisit_of_exhaustive_key_search↩
        ->addWidget(
            std::move(↩
                button_cryptography_symmetric_more_about_block_ciphers_revisit_of_exhaus
                ));
    this->addWidget(std::move(this->↩
        symmetric_more_about_block_ciphers_revisit_of_exhaustive_key_search↩
        ),13,2, Wt::AlignmentFlag::Center);
}

void crypto_online_navigation_grid::setup_asymmetric_contents() {
    this->title_asymmetric = Wt::cpp14::make_unique<Wt::WText>("↩
        Asymmetric Cryptography");
    this->title_asymmetric->setStyleClass("navigation_grid_title");
    this->addWidget(std::move(this->title_asymmetric),14,1);

    this->subtitle_introduction_to_asymmetric_cryptography = Wt::cpp14::↩
        make_unique<Wt::WText>("Introduction to Asymmetric Cryptography");
    this->subtitle_introduction_to_asymmetric_cryptography->setStyleClass↩
        ("navigation_grid_subtitle");
    this->addWidget(std::move(this->↩
        subtitle_introduction_to_asymmetric_cryptography),15,1);

    this->asymmetric_symmetric_vs_asymmetric = Wt::cpp14::make_unique<Wt↩
        ::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/asymmetric/symmetric-vs-↩
                asymmetric"));
    auto button_cryptography_asymmetric_vs_symmetric = Wt::cpp14::↩
        make_unique<Wt::WPushButton>("Symmetric vs Asymmetric");
    button_cryptography_asymmetric_vs_symmetric->setStyleClass("↩
        navigation_grid_item");
    this->asymmetric_symmetric_vs_asymmetric->addWidget(std::move(↩
        button_cryptography_asymmetric_vs_symmetric));
    this->addWidget(std::move(asymmetric_symmetric_vs_asymmetric),16,0, ↩
        Wt::AlignmentFlag::Center);

    this->asymmetric_authenticity_of_public_keys = Wt::cpp14::make_unique↩
        <Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/asymmetric/authenticity-of-↩
                public-keys"));
    auto button_cryptography_asymmetric_authenticity_of_public_keys = Wt↩
        ::cpp14::make_unique<Wt::WPushButton>("Authenticity of Public Keys↩
        ");
    button_cryptography_asymmetric_authenticity_of_public_keys->↩
        setStyleClass("navigation_grid_item");
    this->asymmetric_authenticity_of_public_keys->addWidget(std::move(↩
        button_cryptography_asymmetric_authenticity_of_public_keys));
    this->addWidget(std::move(asymmetric_authenticity_of_public_keys)↩
        ,16,1, Wt::AlignmentFlag::Center);

    this->asymmetric_key_lengths_and_security_levels = Wt::cpp14::↩
        make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/asymmetric/key-lengths-and-↩
                security-levels"));
    auto button_cryptography_asymmetric_key_lengths_security_levels = Wt↩
        ::cpp14::make_unique<Wt::WPushButton>("Key Lengths and Security ↩
        Lengths");
    button_cryptography_asymmetric_key_lengths_security_levels->↩
        setStyleClass("navigation_grid_item");
    this->asymmetric_key_lengths_and_security_levels->addWidget(std::move↩
        (button_cryptography_asymmetric_key_lengths_security_levels));
    this->addWidget(std::move(asymmetric_key_lengths_and_security_levels)↩
,16,2, Wt::AlignmentFlag::Center);

    this->subtitle_basic_number_theory_for_pk = Wt::cpp14::make_unique<Wt::WText>("Basic Number Theory for Public-Key Algorithms");
    this->subtitle_basic_number_theory_for_pk->setStyleClass("navigation_grid_subtitle");
    this->addWidget(std::move(this->subtitle_basic_number_theory_for_pk),17,1);

    this->asymmetric_euclidean_algorithm = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/asymmetric/ea-eea"));
    auto button_cryptography_asymmetric_euclidean_algorithm = Wt::cpp14::make_unique<Wt::WPushButton>("EA and EEA Algorithms");
    button_cryptography_asymmetric_euclidean_algorithm->setStyleClass("navigation_grid_item");
    this->asymmetric_euclidean_algorithm->addWidget(std::move(button_cryptography_asymmetric_euclidean_algorithm));
    this->addWidget(std::move(asymmetric_euclidean_algorithm),18,0, Wt::AlignmentFlag::Center);

    this->asymmetric_eulers_phi_function = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/asymmetric/euler's-phi-function"));
    auto button_cryptography_asymmetric_phi_function = Wt::cpp14::make_unique<Wt::WPushButton>("Euler's Phi Function");
    button_cryptography_asymmetric_phi_function->setStyleClass("navigation_grid_item");
    this->asymmetric_eulers_phi_function->addWidget(std::move(button_cryptography_asymmetric_phi_function));
    this->addWidget(std::move(asymmetric_eulers_phi_function),18,1, Wt::AlignmentFlag::Center);

    this->asymmetric_fermats_little_theorem_and_eulers_theorem = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/asymmetric/fermats-little-theorem-eulers-theorem"));
    auto button_cryptography_fermats_little_theorem_and_eulers_theorem= Wt::cpp14::make_unique<Wt::WPushButton>("Fermats Little Theorem and Eulers Theorem");
    button_cryptography_fermats_little_theorem_and_eulers_theorem->setStyleClass("navigation_grid_item");
    this->asymmetric_fermats_little_theorem_and_eulers_theorem->addWidget(std::move(button_cryptography_fermats_little_theorem_and_eulers_theorem));
    this->addWidget(std::move(asymmetric_fermats_little_theorem_and_eulers_theorem),18,2, Wt::AlignmentFlag::Center);

    this->subtitle_rsa_cryptosystem = Wt::cpp14::make_unique<Wt::WText>("The RSA Cryptosystem");
    this->subtitle_rsa_cryptosystem->setStyleClass("navigation_grid_subtitle");
    this->addWidget(std::move(this->subtitle_rsa_cryptosystem),19,1);

    this->asymmetric_rsa_introduction = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/asymmetric/rsa-introduction"));
    auto button_cryptography_asymmetric_rsa_introduction = Wt::cpp14::make_unique<Wt::WPushButton>("RSA Introduction");
    button_cryptography_asymmetric_rsa_introduction->setStyleClass("navigation_grid_item");
    this->asymmetric_rsa_introduction->addWidget(std::move(button_cryptography_asymmetric_rsa_introduction));
    this->addWidget(std::move(asymmetric_rsa_introduction),20,0, Wt::AlignmentFlag::Center);

    this->asymmetric_rsa_encryption_decryption = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/asymmetric/rsa-encryption-decryption"));
    auto button_cryptography_asymmetric_rsa_encryption_decryption = Wt::cpp14::make_unique<Wt::WPushButton>("RSA Encryption and Decryption");
    button_cryptography_asymmetric_rsa_encryption_decryption->setStyleClass("navigation_grid_item");
    this->asymmetric_rsa_encryption_decryption->addWidget(std::move(button_cryptography_asymmetric_rsa_encryption_decryption));
    this->addWidget(std::move(asymmetric_rsa_encryption_decryption),20,1, Wt::AlignmentFlag::Center);

    this->asymmetric_rsa_key_generation = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/asymmetric/rsa-key-generation"));
    auto button_cryptography_asymmetric_rsa_key_generation = Wt::cpp14::make_unique<Wt::WPushButton>("RSA Key Generation");
    button_cryptography_asymmetric_rsa_key_generation->setStyleClass("navigation_grid_item");
    this->asymmetric_rsa_key_generation->addWidget(std::move(button_cryptography_asymmetric_rsa_key_generation));
    this->addWidget(std::move(asymmetric_rsa_key_generation),20,2, Wt::AlignmentFlag::Center);

    this->asymmetric_rsa_encryption_decryption_fast_exponentiation= Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/asymmetric/rsa-fast-exponentiation"));
    auto button_cryptography_asymmetric_rsa_encryption_decryption_fast_exponentiation =
            Wt::cpp14::make_unique<Wt::WPushButton>("RSA Encryption and Decryption with Fast Exponentiation");
    button_cryptography_asymmetric_rsa_encryption_decryption_fast_exponentiation->setStyleClass("navigation_grid_item");
    this->asymmetric_rsa_encryption_decryption_fast_exponentiation->addWidget(std::move(button_cryptography_asymmetric_rsa_encryption_decryption_fast_exponentiation));
    this->addWidget(std::move(asymmetric_rsa_encryption_decryption_fast_exponentiation),21,0, Wt::AlignmentFlag::Center);

    this->asymmetric_rsa_speed_up_techniques = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
        Wt::LinkType::InternalPath, "/asymmetric/rsa-speed-up-techniques"));
    auto button_cryptography_asymmetric_rsa_speed_up_techniques = Wt::cpp14::make_unique<Wt::WPushButton>("RSA Speed Up Techniques");
    button_cryptography_asymmetric_rsa_speed_up_techniques->setStyleClass("navigation_grid_item");
    this->asymmetric_rsa_speed_up_techniques->addWidget(std::move(button_cryptography_asymmetric_rsa_speed_up_techniques));
    this->addWidget(std::move(asymmetric_rsa_speed_up_techniques),21,1, Wt::AlignmentFlag::Center);

    this->asymmetric_rsa_finding_large_primes = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
        Wt::LinkType::InternalPath, "/asymmetric/rsa-finding-large-primes"));
    auto button_cryptography_asymmetric_rsa_finding_large_primes = Wt::cpp14::make_unique<Wt::WPushButton>("RSA Finding Large Primes");
    button_cryptography_asymmetric_rsa_finding_large_primes->setStyleClass("navigation_grid_item");
    this->asymmetric_rsa_finding_large_primes->addWidget(std::move(button_cryptography_asymmetric_rsa_finding_large_primes));
    this->addWidget(std::move(asymmetric_rsa_finding_large_primes),21,2, Wt::AlignmentFlag::Center);

    this->asymmetric_rsa_padding = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
        Wt::LinkType::InternalPath, "/asymmetric/rsa-padding"));
    auto button_cryptography_asymmetric_rsa_padding = Wt::cpp14::make_unique<Wt::WPushButton>("RSA Padding");
    button_cryptography_asymmetric_rsa_padding->setStyleClass("navigation_grid_item");
    this->asymmetric_rsa_padding->addWidget(std::move(button_cryptography_asymmetric_rsa_padding));
    this->addWidget(std::move(asymmetric_rsa_padding),22,0, Wt::AlignmentFlag::Center);

    this->asymmetric_rsa_attacks = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
        Wt::LinkType::InternalPath, "/asymmetric/rsa-finding-large-primes"));
    auto button_cryptography_asymmetric_rsa_attacks = Wt::cpp14::make_unique<Wt::WPushButton>("RSA Attacks");
    button_cryptography_asymmetric_rsa_attacks->setStyleClass("navigation_grid_item");
    this->asymmetric_rsa_attacks->addWidget(std::move(button_cryptography_asymmetric_rsa_attacks));
    this->addWidget(std::move(asymmetric_rsa_attacks),22,1, Wt::AlignmentFlag::Center);

    this->asymmetric_rsa_implementations_in_hardware_software = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
        Wt::LinkType::InternalPath, "/asymmetric/rsa-finding-large-primes"));
    auto button_cryptography_asymmetric_rsa_implementations_in_hardware_software =
        Wt::cpp14::make_unique<Wt::WPushButton>("RSA Implementations in Hardware and Software");
    button_cryptography_asymmetric_rsa_implementations_in_hardware_software->setStyleClass("navigation_grid_item");
    this->asymmetric_rsa_implementations_in_hardware_software->addWidget(std::move(button_cryptography_asymmetric_rsa_implementations_in_hardware_software));
    this->addWidget(std::move(asymmetric_rsa_implementations_in_hardware_software),22,2, Wt::AlignmentFlag::Center);

    this->subtitle_dlp_cryptosystems = Wt::cpp14::make_unique<Wt::WText>("Public-Key Cryptosystems based on the Discrete Logarithm Problem");
    this->subtitle_dlp_cryptosystems->setStyleClass("navigation_grid_subtitle");
    this->addWidget(std::move(this->subtitle_dlp_cryptosystems),23,1);

    this->asymmetric_dh_dhke = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/asymmetric/dh-dhke"));
    auto button_cryptography_asymmetric_dh_dhke = Wt::cpp14::make_unique<Wt::WPushButton>("Diffie-Hellman Key Exchange");
    button_cryptography_asymmetric_dh_dhke->setStyleClass("navigation_grid_item");
    this->asymmetric_dh_dhke->addWidget(std::move(button_cryptography_asymmetric_dh_dhke));
    this->addWidget(std::move(asymmetric_dh_dhke),24,0, Wt::AlignmentFlag::Center);

    this->asymmetric_dh_algebra= Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/asymmetric/dh-algebra"));
    auto button_cryptography_asymmetric_dh_algebra = Wt::cpp14::make_unique<Wt::WPushButton>("Diffie-Hellman Algebra");
    button_cryptography_asymmetric_dh_algebra->setStyleClass("navigation_grid_item");
    this->asymmetric_dh_algebra->addWidget(std::move(button_cryptography_asymmetric_dh_algebra));
    this->addWidget(std::move(asymmetric_dh_algebra),24,1, Wt::AlignmentFlag::Center);

    this->asymmetric_dh_dlp = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/asymmetric/dh-dlp"));
    auto button_cryptography_asymmetric_dh_dlp = Wt::cpp14::make_unique<Wt::WPushButton>("The Discrete Logarithm Problem");
    button_cryptography_asymmetric_dh_dlp->setStyleClass("navigation_grid_item");
    this->asymmetric_dh_dlp->addWidget(std::move(button_cryptography_asymmetric_dh_dlp));
    this->addWidget(std::move(asymmetric_dh_dlp),24,2, Wt::AlignmentFlag::Center);

    this->asymmetric_dh_security = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/asymmetric/dh-security"));
    auto button_cryptography_asymmetric_dh_security = Wt::cpp14::make_unique<Wt::WPushButton>("Diffie-Hellman Security");
    button_cryptography_asymmetric_dh_security->setStyleClass("↩
        navigation_grid_item");
    this->asymmetric_dh_security->addWidget(std::move(↩
        button_cryptography_asymmetric_dh_security));
    this->addWidget(std::move(asymmetric_dh_security),25,0, Wt::↩
        AlignmentFlag::Center);

    this->asymmetric_dh_elgamal_encryption_scheme = Wt::cpp14::↩
        make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/asymmetric/dh-dlp"));
    auto button_cryptography_asymmetric_dh_encryption_scheme = Wt::cpp14↩
        ::make_unique<Wt::WPushButton>("The Elgamal Encryption Scheme");
    button_cryptography_asymmetric_dh_encryption_scheme->setStyleClass("↩
        navigation_grid_item");
    this->asymmetric_dh_elgamal_encryption_scheme->addWidget(std::move(↩
        button_cryptography_asymmetric_dh_encryption_scheme));
    this->addWidget(std::move(asymmetric_dh_elgamal_encryption_scheme)↩
        ,25,1, Wt::AlignmentFlag::Center);

    this->asymmetric_dh_elgamal_security = Wt::cpp14::make_unique<Wt::↩
        WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/asymmetric/dh-dlp"));
    auto button_cryptography_asymmetric_dh_elgamal_security  = Wt::cpp14↩
        ::make_unique<Wt::WPushButton>("Elgamal Security");
    button_cryptography_asymmetric_dh_elgamal_security->setStyleClass("↩
        navigation_grid_item");
    this->asymmetric_dh_elgamal_security->addWidget(std::move(↩
        button_cryptography_asymmetric_dh_elgamal_security));
    this->addWidget(std::move(asymmetric_dh_elgamal_security),25,2, Wt::↩
        AlignmentFlag::Center);

    this->subtitle_ec_cryptosystems = Wt::cpp14::make_unique<Wt::WText>("↩
        Elliptic Curve Cryptosystems");
    this->subtitle_ec_cryptosystems->setStyleClass("↩
        navigation_grid_subtitle");
    this->addWidget(std::move(this->subtitle_ec_cryptosystems),26,1);

    this->asymmetric_ec_definition = Wt::cpp14::make_unique<Wt::WAnchor>(↩
        Wt::WLink(
            Wt::LinkType::InternalPath, "/asymmetric/ec-definition"));
    auto button_cryptography_asymmetric_ec_definition  = Wt::cpp14::↩
        make_unique<Wt::WPushButton>("Elliptic Curve Definition");
    button_cryptography_asymmetric_ec_definition->setStyleClass("↩
        navigation_grid_item");
    this->asymmetric_ec_definition->addWidget(std::move(↩
        button_cryptography_asymmetric_ec_definition));
    this->addWidget(std::move(asymmetric_ec_definition),27,0, Wt::↩
        AlignmentFlag::Center);

    this->asymmetric_ec_group_operations = Wt::cpp14::make_unique<Wt::↩
        WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/asymmetric/ec-group-operations"↩
                ));
    auto button_cryptography_asymmetric_ec_group_operations = Wt::cpp14::↩
        make_unique<Wt::WPushButton>("Elliptic Curve Group Operations");
    button_cryptography_asymmetric_ec_group_operations->setStyleClass("↩
        navigation_grid_item");
    this->asymmetric_ec_group_operations->addWidget(std::move(↩
        button_cryptography_asymmetric_ec_group_operations));
    this->addWidget(std::move(asymmetric_ec_group_operations),27,1, Wt::↩
        AlignmentFlag::Center);

    this->asymmetric_ec_dlp = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::↩
        WLink(
            Wt::LinkType::InternalPath, "/asymmetric/ec-dlp"));
    auto button_cryptography_asymmetric_ec_dlp  = Wt::cpp14::make_unique<↩
        Wt::WPushButton>("Building a DLP with Elliptic Curves");
    button_cryptography_asymmetric_ec_dlp->setStyleClass("↩
        navigation_grid_item");
    this->asymmetric_ec_dlp->addWidget(std::move(↩
        button_cryptography_asymmetric_ec_dlp));
    this->addWidget(std::move(asymmetric_ec_dlp),27,2, Wt::AlignmentFlag↩
        ::Center);

    this->asymmetric_ec_dhke = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::↩
        WLink(
            Wt::LinkType::InternalPath, "/asymmetric/ec-dhke"));
    auto button_cryptography_asymmetric_ec_dhke = Wt::cpp14::make_unique<↩
        Wt::WPushButton>("DH-Key Exchange with Elliptic Curves");
    button_cryptography_asymmetric_ec_dhke->setStyleClass("↩
        navigation_grid_item");
    this->asymmetric_ec_dhke->addWidget(std::move(↩
        button_cryptography_asymmetric_ec_dhke));
    this->addWidget(std::move(asymmetric_ec_dhke),28,0, Wt::AlignmentFlag↩
        ::Center);

    this->asymmetric_ec_security = Wt::cpp14::make_unique<Wt::WAnchor>(Wt↩
        ::WLink(
            Wt::LinkType::InternalPath, "/asymmetric/ec-security"));
    auto button_cryptography_asymmetric_ec_security = Wt::cpp14::↩
        make_unique<Wt::WPushButton>("Elliptic Curve Security");
    button_cryptography_asymmetric_ec_security->setStyleClass("↩
        navigation_grid_item");
    this->asymmetric_ec_security->addWidget(std::move(↩
        button_cryptography_asymmetric_ec_security));
    this->addWidget(std::move(asymmetric_ec_security),28,1, Wt::↩
        AlignmentFlag::Center);

    this->asymmetric_ec_implementations_in_hardware_software= Wt::cpp14::↩
        make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/asymmetric/ec-implementations")↩
                );
    auto ↩
        button_cryptography_asymmetric_ec_implementations_in_hardware_software↩
          =
            Wt::cpp14::make_unique<Wt::WPushButton>("Elliptic Curve ↩
                Implementations in Hardware and Software");
    button_cryptography_asymmetric_ec_implementations_in_hardware_software↩
        ->setStyleClass("navigation_grid_item");
    this->asymmetric_ec_implementations_in_hardware_software->addWidget(↩
        std::move(↩
        button_cryptography_asymmetric_ec_implementations_in_hardware_software↩
        ));
    this->addWidget(std::move(↩
        asymmetric_ec_implementations_in_hardware_software),28,2, Wt::↩
        AlignmentFlag::Center);

    this->subtitle_digital_signatures = Wt::cpp14::make_unique<Wt::WText↩
        >("Digital Signatures");
    this->subtitle_digital_signatures->setStyleClass("↩
        navigation_grid_subtitle");
    this->addWidget(std::move(this->subtitle_digital_signatures),29,1);

    this->asymmetric_ds_introduction= Wt::cpp14::make_unique<Wt::WAnchor↩
        >(Wt::WLink(
              Wt::LinkType::InternalPath, "/asymmetric/ds-introduction"));
    auto button_cryptography_asymmetric_ds_introduction = Wt::cpp14::↩
        make_unique<Wt::WPushButton>("Introduction to Digital Signatures")↩
        ;
    button_cryptography_asymmetric_ds_introduction->setStyleClass("↩
        navigation_grid_item");
    this->asymmetric_ds_introduction->addWidget(std::move(↩
        button_cryptography_asymmetric_ds_introduction));
    this->addWidget(std::move(asymmetric_ds_introduction),30,0, Wt::↩
        AlignmentFlag::Center);

    this->asymmetric_ds_rsa = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::↩
        WLink(
              Wt::LinkType::InternalPath, "/asymmetric/ds-rsa"));
    auto button_cryptography_asymmetric_ds_rsa = Wt::cpp14::make_unique<↩
        Wt::WPushButton>("RSA Digital Signatures");
    button_cryptography_asymmetric_ds_rsa->setStyleClass("↩
        navigation_grid_item");
    this->asymmetric_ds_rsa->addWidget(std::move(↩
        button_cryptography_asymmetric_ds_rsa));
    this->addWidget(std::move(asymmetric_ds_rsa),30,1, Wt::AlignmentFlag↩
        ::Center);

    this->asymmetric_ds_elgamal = Wt::cpp14::make_unique<Wt::WAnchor>(Wt↩
        ::WLink(
              Wt::LinkType::InternalPath, "/asymmetric/ds-elgamal"));
    auto button_cryptography_asymmetric_ds_elgamal = Wt::cpp14::↩
        make_unique<Wt::WPushButton>("Elgamal Digital Signatures");
    button_cryptography_asymmetric_ds_elgamal->setStyleClass("↩
        navigation_grid_item");
    this->asymmetric_ds_elgamal->addWidget(std::move(↩
        button_cryptography_asymmetric_ds_elgamal));
    this->addWidget(std::move(asymmetric_ds_elgamal),30,2, Wt::↩
        AlignmentFlag::Center);

    this->asymmetric_ds_dsa = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::↩
        WLink(
              Wt::LinkType::InternalPath, "/asymmetric/ds-dsa"));
    auto button_cryptography_asymmetric_ds_dsa = Wt::cpp14::make_unique<↩
        Wt::WPushButton>("Digital Signature Algorithm");
    button_cryptography_asymmetric_ds_dsa->setStyleClass("↩
        navigation_grid_item");
    this->asymmetric_ds_dsa->addWidget(std::move(↩
        button_cryptography_asymmetric_ds_dsa));
    this->addWidget(std::move(asymmetric_ds_dsa),31,0, Wt::AlignmentFlag↩
        ::Center);

    this->asymmetric_ds_ecdsa = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::↩
        WLink(
            Wt::LinkType::InternalPath, "/asymmetric/ec-security"));
    auto button_cryptography_asymmetric_ds_ecdsa = Wt::cpp14::make_unique<Wt::WPushButton>("Elliptic Curve Digital Signatures");
    button_cryptography_asymmetric_ds_ecdsa->setStyleClass("navigation_grid_item");
    this->asymmetric_ds_ecdsa->addWidget(std::move(button_cryptography_asymmetric_ds_ecdsa));
    this->addWidget(std::move(asymmetric_ds_ecdsa),31,2, Wt::AlignmentFlag::Center);

}

void crypto_online_navigation_grid::setup_applications_contents() {
    this->title_applications = Wt::cpp14::make_unique<Wt::WText>("Protocols");
    this->title_applications->setStyleClass("navigation_grid_title");
    this->addWidget(std::move(this->title_applications),32,1);

    this->subtitle_hash_functions = Wt::cpp14::make_unique<Wt::WText>("Hash Functions");
    this->subtitle_hash_functions->setStyleClass("navigation_grid_subtitle");
    this->addWidget(std::move(this->subtitle_hash_functions),33,1);

    this->protocols_hf_motivation = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/protocols/hash-function-motivation"));
    auto button_cryptography_protocol_hf_motivation = Wt::cpp14::make_unique<Wt::WPushButton>("Why we need Hash Functions");
    button_cryptography_protocol_hf_motivation->setStyleClass("navigation_grid_item");
    this->protocols_hf_motivation->addWidget(std::move(button_cryptography_protocol_hf_motivation));
    this->addWidget(std::move(this->protocols_hf_motivation),34,0, Wt::AlignmentFlag::Center);

    this->protocols_hf_security_requirements = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/protocols/hash-function-security-requirements"));
    auto button_cryptography_protocol_hf_security_requirements = Wt::cpp14::make_unique<Wt::WPushButton>("Security Requirements");
    button_cryptography_protocol_hf_security_requirements->setStyleClass("navigation_grid_item");
    this->protocols_hf_security_requirements->addWidget(std::move(button_cryptography_protocol_hf_security_requirements));
    this->addWidget(std::move(this->protocols_hf_security_requirements),34,1, Wt::AlignmentFlag::Center);

    this->protocols_hf_overview = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/protocols/hash-function-overview"));
    auto button_cryptography_protocol_hf_overview = Wt::cpp14::make_unique<Wt::WPushButton>("Hash Function Overview");
    button_cryptography_protocol_hf_overview->setStyleClass("navigation_grid_item");
    this->protocols_hf_overview->addWidget(std::move(button_cryptography_protocol_hf_overview));
    this->addWidget(std::move(this->protocols_hf_overview),34,2, Wt::AlignmentFlag::Center);

    this->protocols_hf_sha1 = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/protocols/hash-function-sha1"));
    auto button_cryptography_protocol_hf_sha1 = Wt::cpp14::make_unique<Wt::WPushButton>("SHA-1");
    button_cryptography_protocol_hf_sha1->setStyleClass("navigation_grid_item");
    this->protocols_hf_sha1->addWidget(std::move(button_cryptography_protocol_hf_sha1));
    this->addWidget(std::move(this->protocols_hf_sha1),35,0, Wt::AlignmentFlag::Center);

    this->protocols_hf_sha3 = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/protocols/hash-function-overview"));
    auto button_cryptography_protocol_hf_sha3=  Wt::cpp14::make_unique<Wt::WPushButton>("SHA-3");
    button_cryptography_protocol_hf_sha3->setStyleClass("navigation_grid_item");
    this->protocols_hf_sha3->addWidget(std::move(button_cryptography_protocol_hf_sha3));
    this->addWidget(std::move(this->protocols_hf_sha3),35,1, Wt::AlignmentFlag::Center);

    this->protocols_hf_bcrypt = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/protocols/hash-function-overview"));
    auto button_cryptography_protocol_hf_bcrypt = Wt::cpp14::make_unique<Wt::WPushButton>("bcrypt");
    button_cryptography_protocol_hf_bcrypt->setStyleClass("navigation_grid_item");
    this->protocols_hf_bcrypt->addWidget(std::move(button_cryptography_protocol_hf_bcrypt));
    this->addWidget(std::move(this->protocols_hf_bcrypt),35,2, Wt::AlignmentFlag::Center);

    this->subtitle_macs = Wt::cpp14::make_unique<Wt::WText>("Message Authentication Codes");
    this->subtitle_macs->setStyleClass("navigation_grid_subtitle");
    this->addWidget(std::move(this->subtitle_macs),36,1);

    this->protocols_macs_principles = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/protocols/mac-principles"));
    auto button_cryptography_protocol_mac_principles =
            Wt::cpp14::make_unique<Wt::WPushButton>("Message Authentication Code Principles");
    button_cryptography_protocol_mac_principles->setStyleClass("navigation_grid_item");
    this->protocols_macs_principles->addWidget(std::move(
        button_cryptography_protocol_mac_principles));
    this->addWidget(std::move(this->protocols_macs_principles),37,0, Wt::AlignmentFlag::Center);

    this->protocols_macs_hash_functions= Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/protocols/mac-hmac"));
    auto button_cryptography_protocol_macs_hash_functions =
            Wt::cpp14::make_unique<Wt::WPushButton>("Message Authentication Codes From Hash Functions");
    button_cryptography_protocol_macs_hash_functions->setStyleClass("navigation_grid_item");
    this->protocols_macs_hash_functions->addWidget(std::move(button_cryptography_protocol_macs_hash_functions));
    this->addWidget(std::move(this->protocols_macs_hash_functions),37,1, Wt::AlignmentFlag::Center);

    this->protocols_macs_block_cipher = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/protocols/mac-cbcmac"));
    auto button_cryptography_protocol_mac_block_cipher =
            Wt::cpp14::make_unique<Wt::WPushButton>("Message Authentication Codes From Block Ciphers");
    button_cryptography_protocol_mac_block_cipher->setStyleClass("navigation_grid_item");
    this->protocols_macs_block_cipher->addWidget(std::move(button_cryptography_protocol_mac_block_cipher));
    this->addWidget(std::move(this->protocols_macs_block_cipher),37,2, Wt::AlignmentFlag::Center);

    this->subtitle_key_establishment = Wt::cpp14::make_unique<Wt::WText>("Key Establishment");
    this->subtitle_key_establishment->setStyleClass("navigation_grid_subtitle");
    this->addWidget(std::move(this->subtitle_key_establishment),38,1);

    this->protocols_key_est_introduction = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/protocols/key-est-introduction"));
    auto button_cryptography_protocol_key_est_introduction =
            Wt::cpp14::make_unique<Wt::WPushButton>("Introduction to Key Establishment");
    button_cryptography_protocol_key_est_introduction->setStyleClass("navigation_grid_item");
    this->protocols_key_est_introduction->addWidget(std::move(button_cryptography_protocol_key_est_introduction));
    this->addWidget(std::move(this->protocols_key_est_introduction),39,0, Wt::AlignmentFlag::Center);

    this->protocols_key_est_symmetric = Wt::cpp14::make_unique<Wt::WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/protocols/key-est-symmetric"));
    auto button_cryptography_protocol_key_est_symmetric =
            Wt::cpp14::make_unique<Wt::WPushButton>("Key Establishment with Symmetric Techniques");
    button_cryptography_protocol_key_est_symmetric->setStyleClass("navigation_grid_item");
    this->protocols_key_est_symmetric->addWidget(std::move(↩
       button_cryptography_protocol_key_est_symmetric));
    this->addWidget(std::move(this->protocols_key_est_symmetric),39,1, Wt↩
       ::AlignmentFlag::Center);

    this->protocols_key_est_asymmetric = Wt::cpp14::make_unique<Wt::↩
       WAnchor>(Wt::WLink(
            Wt::LinkType::InternalPath, "/protocols/key-est-asymmetric"))↩
                ;
    auto button_cryptography_protocol_key_est_asymmetric =
            Wt::cpp14::make_unique<Wt::WPushButton>("Key Establishment ↩
                with Asymmetric Techniques");
    button_cryptography_protocol_key_est_asymmetric->setStyleClass("↩
       navigation_grid_item");
    this->protocols_key_est_asymmetric->addWidget(std::move(↩
       button_cryptography_protocol_key_est_asymmetric));
    this->addWidget(std::move(this->protocols_key_est_asymmetric),39,2, ↩
       Wt::AlignmentFlag::Center);

}

void crypto_online_navigation_grid::clear_grid() {

}
```

Chapter 4

Testing

Chapter 5

Evaluation

List of Figures